LA PROGRAMLINGVO

PASCAL

(NOVEMBRO 1982)

CHRISTIAN BERTIN

LA PROGRAMLINGVO

PASCAL

(NOVEMBRO 1982)

Redaktoro :

Christian BERTIN
CCETT AIS
B.P : 59
Rue du Clos Courtel
F-35510 CESSON-SEVIGNE
FRANCIO

Antaŭparolo

La programadlingvo PASCAL (elp. PASKAL) estas unu el la plej novaj programadlingvoj. Tamen pli kaj pli da komputorfabrikistoj proponas kompilerojn por ĝi.

Tiu nova programadlingvo aplikiĝas por solvi teknikajn kaj sciencajn problemojn sed ankaŭ por solvi administrajn problemojn.

Plie ĝi estas pli kaj pli uzata en informadikaj projektoj, precipe kiam la farota softvaro estas tre ampleksa (ĝi estas uzata kaj por la programaro kaj por la dokumentoj kiuj klarigas la programaron).

Tiu programadlingvo jam pli uziĝas ol ALGOL.

Tiu kajero prezentas la programadlingvon PASCAL: ties historion kaj ties karakterizojn, al tio aldoniĝas kelkaj ekzemploj de programoj skribitaj en PASCAL.

I. HISTORIO DE LA PROGRAMLINGVO "PASCAL".

I.1. La unuaj paŝoj.

La unua versio de la programlingvo PASCAL (elp.: paskal) estas verkita de Prof. Niklaus Wirth en la Eidgenossiche Technische Hochshule en Zurich, en Svisio en 1968, surbaze de la lingva spirito de ALGOL 60 kaj ALGOL-W. Tion li faris por respondi al du ĉefaj bezonoj :

- disponi pri programlingvo adekvata por instrui klaran kaj logikan programadon.
- difini programlingvon kies uzo estu fidinda kaj efika en la nunaj komputoroj.

La unua kompilero de Wirth ekfunkciis en 1970.

Pro la kreskanta interesiĝo en la uzo de kompileroj por aliaj komputoroj kaj por enkonduki kelkajn pliboniĝojn en la lingvon, aktualigita "raporto" estis publikigita en 1973.

I.2. Kontribuo de la Universitato de Kalifornio en San Diego.

La Instituto pri InformSistemoj de la Universitato de Kalifornio en San Diego (UCSD), sub la direktado de Kenneth Bowles, estis la ĉefa aganto por la disvastigo de Pascal en Usono. En 1974, Bowles ekinteresiĝis al Pascal pro ĝiaj avantaĝoj en struktura programado por la instruado, sed li ankaŭ volis transporteblan lingvon, tio estas : lingvo ne ligita al iu aparta komputoro. Per frekvencbaza kodado, li runigis la kompileron en PDP-11 kaj nelonge poste, en iu mikroprocesoro. Tiam, li kaj liaj kolegoj aldonis unuuzantan opson kiu jam vaste estis uzata.

Iom poste, aliaj kompileroj aperis kun diversaj apartaj aldonaĵoj. La normiga bezono pri tiuj aldonaĵoj agnoskiĝis de la paskalaj advokatoj kaj, en la somero de 1978, laborkunvenon gastigis la UCSDa Instituto pri InformSistemoj. El tiu kunveno eliris areto da aldonaĵoj kun vasta industria konsento. La aldonaĵoj akceptitaj havas la sekvantajn karakterizojn :

a) akordebleco : ili ne eksvalidigas programojn skribitajn en la origina lingvo.

b) taŭgeco kaj neceseco : aldonaĵo karakteriziĝas aŭ per absoluta neceseco por la celita aplik-kampo, aŭ per altgrada taŭgeco kiam komparata al la lingvo sen tiu aldonaĵo.

c) implementebleco : aldonaĵo estas normigita nur se iu implemento povis esti proponata kiu ne kondukis al ekscesa tradukila komplikeco, aŭ malavantaĝoj pro plia runtempo aŭ pro instrukcikvanto.

Nun, la Usona Nacia Norma Instituto (ANSI: American National Standards Institute) kaj la Instituto de la Elektraj kaj Elektronikaj Inĝenieroj (IEEE: Institute of Electrical and Electronics Engineers) konsentis komune studi normon por la programlingvo Pascal.

II. PRISKRIBO DE LA ELEMENTOJ DE LA PROGRAMLINGVO PASCAL.

II.1. Antaŭdifinitaj lingvaj elementoj.

II.1.1. Bazaj simboloj.

La bazaj simboloj konsistas el ĉiuj majusklaj kaj minusklaj literoj, el ĉiuj ciferoj, el la substreko, el la spaceto kaj el la sekvantaj simboloj prezentataj inter duoblaj apostrofoj :

"+", "-", "*", "/", "=", "<", ">", "(", ")", "[", "]", ".", ",", ";",
":", "'", "@", "{", "}", "↑"; " ", "#", "<>", "<=", ">=", ":=", "&",
"!", "(.", ".)", "(*", "*)", "%", ".."

Kutime, por ĉiu aparta kompilero, nur iu grupo de tiuj specialaj simboloj havas apartan signifon. Por ekzemplo :

- "#" kaj "<>" signifas "malsamas".
- "(." kaj ".)" havas la saman signifon kiel "[" kaj "]" (respektive) por la tajpiloj kiuj ne posedas la lastajn.
- "(*" kaj "*)" havas la saman signifon kiel "{" kaj "}" (respektive) por la tajpiloj kiuj ne posedas la lastajn. Ankaŭ "%" estas akceptata por ambaŭ simboloj de iuj kompileroj.
- "@" havas la saman signifon kiel "↑" por la tajpiloj kiuj ne posedas la lastan.
- "!" havas la saman signifon kiel la rezervita vorto "AU" por iuj kompileroj.
- "&" havas la saman signifon kiel la rezervita vorto "KAJ" por iuj kompileroj.

II.1.2. Rezervitaj vortoj.

Tiuj vortoj kiuj konsistas nur el literoj ne povas esti uzataj kiel identigiloj. La listo de tiuj rezervitaj vortoj troviĝas en anekso 1.

Kutime ĉiuj vortoj (kaj precipe la rezervitaj vortoj) estas skribitaj aŭ nur majuskle aŭ nur minuskle, sed por faciligi la distingon inter la rezervitaj vortoj kaj la aliaj vortoj, en tiu kajero ni preferis uzi majusklojn nur por la rezervitaj vortoj kaj rezervitaj identigiloj kaj minusklojn nur por la aliaj.

II.1.3. Apartigiloj.

Spaceto(j), linifino(j) kaj komentario(j) estas apartigiloj. Neniu apartigilo devas troviĝi ene de identigilo, de nombro, de rezervita vorto.

Almenaŭ unu apartigilo devas troviĝi inter du sinsekvaj identigiloj, nombroj aŭ rezervitaj vortoj.

II.2. Lingvaj elementoj difinotaj de la uzanto.

II.2.1. La identigiloj.

Ili konsistas el grupo da majusklaj aŭ minusklaj litero(j), substreko(j) kaj cifero(j).

Ili komenciĝas per litero aŭ substreko, povas esti unukaraktraj aŭ plurkaraktraj sed ne povas esti unu el la rezervitaj vortoj. Ili povas esti tre longaj sed depende de la kompileroj nur la 8 aŭ 10 unuaj karaktroj estas konsiderataj, konsekvence du identigiloj estas identaj se estas neniu diferenco inter ili en la 8 aŭ 10 unuaj karaktroj.

Jen ekzemploj de validaj identigiloj :

_ s fin12 abio salti_al

II.2.2. La nombroj.

La entjeraj nombroj konsistas el eventuala signo (-) aŭ ( ), el eventuala indiko de la notacia bazo (ĉeviga se la bazo ne estas 10) kaj el grupo da karaktroj uzataj por la nombroj en tiu notacia bazo.

Jen ekzemploj de validaj entjeraj nombroj :

426 -5 16#E59C -8#72 2#1010

"E59C" estas deksesuma nombro.

"72" estas okuma nombro kaj "1010" estas duuma nombro.

Iuj kompileroj akceptas deksesumajn nombrojn tiel :

'E59C'X anstataŭ la regula : 16#E59C

La frakciaj nombroj enhavas punkton por apartigi la entjeran parton de la frakcia kaj la literon "E" por indiki la potencon.

Jen ekzemploj de validaj frakciaj nombroj :

2.5 -0.3 86.124E+2 37.2E-11

II.2.3. La ĉenoj.

La ĉenoj konsistas el linia grupo da iuj ajn videblaj ASCIIaj karaktroj limigitaj per apostrofo ĉe ĉiu fino. Por inkluzivigi apostrofon en ĉenon, oni duobligas ĝin.

Jen ekzemploj de validaj ĉenoj :

'T' ' ' 'a-5' 'DANK''AL' 'Tio estas ĉeno'

II.2.4. La komentarioj.

La komentarioj konsistas el iu ajn grupo da videblaj ASCIIaj karaktroj troviĝanta inter "(*" kaj "*)".

Jen validaj ekzemploj de komentarioj :

(* Tio estas komentario de la programisto *)

(* subtraho farita, ni videbligu la rezulton *)

Iuj kompileroj akceptas aliajn komentariajn limigilojn, por ekzemplo :

aŭ limigilo "%" : % jen komentario %

aŭ limigiloj "{" kaj "}" : { jen alia komentario }

Ekde nun ni uzos la limigilon "%".

III. PROGRAMO EN LINGVO PASCAL.

PASKALa programo konsistas el :

- program-kapumo.
- deklaroj, difinoj kaj initoj (unua blokparto).
- instrukcifrazoj (dua blokparto).
- program-vostumo (kiu reprezentiĝas per punkto).

III.1. La program-kapumo.

La programkapumo konsistas el la rezervita vorto "PROGRAM" sekvata de identigilo (nomo de la programo) kaj eventuale de listo de identigiloj interkrampe (listo de la parametroj); la punktokomo indikas la finon de tiu programkapumo.

Jen ekzemploj de validaj program-kapumoj :

```
PROGRAM hazardo;
PROGRAM sumo (a,b,rez);
```

III.2. Deklaroj, difinoj kaj initoj (unua blokparto).

Tiu programparto konsistas el :

- deklaro de etikedoj.
- difino de konstantoj.
- difino de tipoj.
- deklaro de variabloj.

- inito de variabloj (nur en la ĉefa bloko).
- deklaro de subrutinoj.
- deklaro de funkcioj.

Iuj kompileroj postulas tiun ordon en la diversaj deklaroj, difinoj kaj initoj.

III.2.1. Deklaro de la etikedoj.

Tiu parto komenciĝas per la rezervita vorto "ETIK" sekvata de listo de etikedoj konsistantaj el nombroj, vortoj aŭ grupo da literoj kaj ciferoj.

Jen ekzemplo de valida deklaro de etikedoj:

```
ETIK 1990, test, vok, pet_1, 16#A2C;
```

Iuj kompileroj akceptas nur dekumajn nombrojn kiel etikedojn kaj/aŭ limigas ilian longon (je 4 por ekzemplo).

III.2.2. Difino de konstantoj.

Tiu parto komenciĝas per la rezervita vorto "KONST" sekvata de listo de identigiloj al kiuj oni atribuas valoron.

Jen ekzemplo de valida difino de konstantoj :

```
KONST supro=10, e=2.71828, vorto='nulo';
```

III.2.3. Difino de tipoj.

Datentipo difiniĝas per la valoroj kiujn variabloj de tiu tipo povas havi, konsekvence simpla maniero por difini datentipon estas listigi ĝiajn valorojn.

III.2.3.1. Tipo "skalaro".

Skalara tipo difiniĝas per orda aro de valoroj, konsekvence difinon de skalara tipo ni povas fari per listigo de la identigiloj kiuj reprezentas tiujn valorojn.

Por ekzemplo:

```
  TIP kolor = (flava,verda,blua,blanka);
      form = (cirklo, kvadrato, kruco);
      valut = (dolaro, pundo, marko, guldeno);
      sezon = (vintr, printemp, somer, aŭtun);
      besto = (kuniklo, koko, anaso, ansero);
```

Iuj skalaraj tipoj estas jam konataj de la kompileroj, temas pri la tipoj : entjero, buleaĵo, karaktro kaj reelo, pro tio ne necesas difini ilin.

ENT: Variablo de tiu tipo povas valori iun ajn entjeran valoron.

Por ekzemplo: -4 -127 0 3624 +10231

BUL: Variablo de tiu tipo povas valori aŭ VERA aŭ FALSA.

KAR: Variablo de tiu tipo povas havi la valoron de iu ajn videbla karaktro.

Por ekzemplo: A s + " 8 _

REEL: Variablo de tiu tipo povas havi iun ajn valoron de reelo.

Por ekzemplo: 3.05 -0.01 +12.37E-5 -237.1

III.2.3.2. Tipo "intervalo".

Intervala tipo difiniĝas kiel parto de alia jam difinita skalara tipo (escepte de la tipo REEL) per indiko de la plej malaltranga kaj de la plej altranga valoroj kiuj limigas tiun parton.

La unua valoro precizigas la malsupran limon de la intervalo kaj konsekvence ne devas superi la supran limon indikatan de la dua valoro, du apudaj punktoj apartigas la du valorojn.

Por ekzemplo (el la tipoj antaŭe difinitaj):

```
  TIP indic = 1..25;
      eŭropvalut = pundo..guldeno;
      bird = koko..ansero;
      jar = 7..77;
```

III.2.3.3. Tipo "tabelo".

Tabela tipo estas strukturo kiu konsistas el fiksita nombro de elementoj de la sama tipo, tiu strukturo estas homogena. Iu ajn elemento de tabelo estas atingebla per indico; la indicoj estas valoroj apartenantaj al alia tipo: la tipo "indico". La indica tipo devas esti skalara tipo aŭ intervala tipo (tio estas:la indicoj ne povas esti reeloj).

La difino de tabela tipo entenas informon pri la elementoj kaj la indico.

La rezervita vorto por la tabela tipo estas "TAB" (mallongigo de TABelo).

Por ekzemplo:

```
TIP no = TAB (.1..15.) DE ENT;
    damtabul = TAB (.1..10,1..10.) DE 0..1;
    vektor = TAB (.1..n.) DE REEL;
```

Kun malpli granda prilabora efikeco, ni povas malgrandigi la memoron necesan por tabelo per aldono de la rezervita vorto "KOMPA" (mallongigo por KOMPaktA).

Por ekzemplo:

```
TIP nb = (.1..15.);
VAR tl: KOMPA TAB (.200.) DE nb;
```

La tipo "nb" enhavas nur 15 elementojn, ni povas kodi ilin per 4 bitoj, tiel la variablo "tl" kiu estas difinita kiel tabelo el 200 elementoj okupos nur 800 bitojn. Se la vortoj de la komputoro estas 32-bitaj, 25 vortoj sufiĉos; sen la rezervita vorto "KOMPA", ni bezonus 200 vortojn.

Unu tabela tipo estas jam konata de la kompileroj, t.e. la tipo "ALFA" kies difino estas:

```
TIP ALFA=KOMPA TAB (.1..n.) DE KAR;
```

kie "n", kiu dependas de la kompileroj, povas esti 4, 8, 16 aŭ k.t.p.

III.2.3.4. Tipo "rikordo".

Rikordo estas strukturo kiu konsistas el fiksita nombro de elementoj nomataj kampoj. Tiuj elementoj povas esti ne samtipaj; tial por ĉiu kampo de rikordo necesas difini ĝian identigilon kaj ĝian tipon.

La rezervita vorto por la rikorda tipo estas "RIK" (mallongigo de RIKordo).

Por ekzemplo:

```
TIP dat = RIK tag: 1..31;
              monat: 1..12;
              jar: 0..5000
          FIN;
```

```
UL = RIK nom: TAB(.1..10.) DE KAR; % nomo %
         anom: TAB(.1..10.) DE KAR; % antaŭnomo %
         nasktag: dat; % naskiĝdato %
         seks: (vira, ina); % sekso %
         famstat: (fraŭl, edz, vidv, divorc); % familia stato %
         profesi: TAB(.1..10.) DE KAR
    FIN;
```

Vi certe rimarkis ke tiu lasta rikordo "ul" entenas alian rikordon : "dat".

Rikordo, krom sia listo de fiksitaj kampoj, povas havi varian parton komenciĝantan per kampselektilo aŭ ne, sekvata de valoroj kiujn ĝi povas havi, uzataj kiel etikedoj al kamplistoj nomataj variantoj.

Tiel diversaj variabloj de la sama rikorda tipo povas havi iom malsaman strukturon. La tipo de la kampselektilo estas skalara (sed ne reela).

Por ekzemplo:

```
TIP stat = (fraŭl, edz, vidv, divorc);
    ul = RIK nom: TAB(.1..10.) DE KAR;
         KAZ stat INTER
             fraŭl: ();
             edz: (edzdat: dat); %"dat": antaŭe difinita tipo %
             vidv: (vidvdat: dat);
             divorc: (divdat: dat, unudiv: BUL) %unua divorco ? %
         FIN
```

Se la kampselektilo enhavas kampidentigilon, ĝi estos konsiderata kiel kampo simila al la aliaj kampoj.

La antaŭa ekzemplo enhavas kampselektilon sen kampidentigilo en la varia parto, tion ni povas ŝanĝi tiel:

```
    . . . . . . .
        KAZ fstat: stat INTER
            ......
```

III.2.3.5. Tipo "aro".

Ara tipo difinas kolekton de valoroj kiu konsistigas la plej grandan eblan enhavon de la variabloj de tiu tipo. Tio estas : variablo samtempe povas havi plurajn valorojn.

La rezervita vorto por la ara tipo estas "AR DE".

Por ekzemplo:

```
TIP kolor = (blanka, blua, verda, ruĝa, nigra, flava);
    flag = AR DE kolor; % flagoj estas plurkoloraj aŭ unukoloraj %
```

Klarigo :

- variablo de la skalara tipo "kolor" povas valori aŭ "blanka" aŭ "blua" aŭ "verda" aŭ ...

- variablo de la ara tipo "flag" povas valori aŭ nenio () aŭ "blanka" aŭ "blanka" kaj "blua" aŭ "blanka" kaj "verda" kaj "nigra" aŭ ....

Aliaj ekzemploj:

```
TIP liter = ('A'..'Z');
    alfabet = AR DE liter;
    nombrovic = AR DE ENT;
```

III.2.3.6. Tipo "rikordaro".

Rikordara tipo difiniĝas kiel sinsekvo de samtipaj elementoj, sed nur unu el tiuj elementoj estas tuj atingebla je iu momento (malsame al tabela tipo); la aliaj elementoj estas atingeblaj nur per paŝpostpaŝa moviĝo tra la rikordaro. Aliflanke la nombro de la elementoj ne estas difinita kaj povas esti nulo: tiakaze la rikordaro estas malplena.

La rezervita vorto por la rikordara tipo estas "RIKAR" (mallongigo de RIKordARo).

Por ekzemplo:

```
TIP ul = RIK nom: TAB(.1..10.) DE KAR;
             numer: ENT
         FIN;
    ulrikar = RIKAR DE ul; % ula rikordaro %
    primrikar = RIKAR DE ENT; % primnombra rikordaro %
```

Rikordaro kies elementoj estas karaktroj ("KAR") nomiĝas "teksta rikordaro", ĝi estas jam konata de la kompileroj, ne necesas difini ĝin en programoj, la rezervita vorto por tiu rikordara tipo estas "TEKST".

La tekstrikordara tipo estas jam difinita tiel:

```
TIP TEKST  RIKAR DE KAR;
```

Tekstaj rikordaroj estas dividitaj laŭ linioj kies finon indikas karaktro ne apartenanta al la teksto mem; tiu karaktro povas esti generata kaj rekonata per apartaj operatoroj.

III.2.3.7. Tipo "ĉeno".

Variablo de ĉena tipo povas valori karaktran ĉenon. La ĉenoj longas je 1 ĝis valoro kiun ĝi ricevas dum la efektivigo de la programo. Por ĉiu ĉena tipo estas difinita maksimuma longo kiun la variabloj de tiu tipo ne povos superi.

La rezervita vorto por la ĉena tipo estas "CHEN".

Por ekzemplo:

```
TIP nom = CHEN(.10.);
```

Se la karaktroj [kaj] ekzistas, ili povas anstataŭi la simbolojn (. kaj .) por simpligo.

III.2.3.8. Tipo "montrilo".

Variablo estas "statika" kiam ĝi estas deklarita en programo, do utiligebla per sia identigilo. Tiu variablo ekzistas nur dum la tuta efektivigo de la bloko en kiu ĝi estas deklarita.

Sed, variabloj povas esti "dinamikaj", t.e. dinamike generataj sen rilato kun la statika strukturo de programo. Per la subrutino "NOV", ni povas generi ilin; ĉar ili ne aperas en variabldeklara parto, ni ne povas utiligi ilin per variabla identigilo. Anstataŭe, ni atingas ilin per montrila variablo kiu ricevas valoron generante dinamikan variablon.

Tiel, montrila tipo konsistas el nefinia aro de valoroj (entjeraj) montrantaj al samtipaj elementoj, inter tiuj valoroj troviĝas "NIL" kiu montras al nenio.

Por ekzemplo:

```
TIP par = RIK a,b: TAB(.1..10.) DE KAR;
          FIN;
    ref= @par;
```

Kun montrilaj variabloj, ni povas konstrui kompleksajn strukturojn, se dinamika variablo enhavas deklaron de pluraj montrilaj variabloj.

Por ekzemplo:

```
TIP ligil = @pers;
    pers = RIK nom: CHEN (.10.);
                post: ligil
           FIN
```

Rimarko: Tio estas la nura kazo en kiu ni povas uzi identigilon (ĉi tie "pers") sen antaŭa difino.

III.2.4. Deklaro de variabloj.

Iu ajn variablo kiu aperas en instrukcio devas esti antaŭe deklarita en la variablodeklara parto. Variablodeklaro asociigas identigilon al variablo-tipo. La rezervita vorto "VAR" enkondukas tiun sekcion.

Por ekzemplo:

```
VAR i,j : ENT; a : KAR; jesne: BUL; frak: REEL;
```

Per tio ni deklaras du entjerajn variablojn "i" kaj "j", unu karaktran variablon "a", unu bulean variablon "jesne" kaj unu reelan variablon "frak".

Jen aliaj ekzemploj de variablodeklaroj rilataj al antaŭe difinitaj tipoj :

```
VAR farb: kolor; bild: form; i: indic;
    aĝo: jar; lud: damtabul; vort: ALFA;
    hodi: dat; pers: ul; landf: flag; dentistar: ulrikar;
    libr: TEKST; var0: nom; k,l: ref;
```

En tiuj ekzemploj, ĉiu linio entenas deklarojn de variabloj kies tipoj ni antaŭe difinis en la tipdifina parto. Tiel ni deklaras "farb" kun skalara tipo "kolor", tio signifas ke "farb" povos havi nur la valorojn "flava", "verda", "blua" aŭ "blanka".

Ni deklaras "bild" kun skalara tipo "form", tio signifas ke "bild" povos havi je iu ajn momento nur unu el la valoroj : "cirklo", "kvadrato" aŭ "kruco".

Ni deklaras "i" kun intervala tipo "indic", tio signifas ke "i" povos havi nur entjeran valoron de 1 ĝis 25.

Ni deklaras "aĝo" kun intervala tipo "jar", tio signifas ke "aĝo" povos havi nur entjeran valoron de 7 ĝis 77.

Ni deklaras "lud" kun tabela tipo "damtabul", tio signifas ke "lud" estas tabelo kies dimensioj estas 10x10 kaj kies elementoj valoras aŭ 0 aŭ 1.

Ni deklaras "vort" kun tabela tipo ALFA, tio signifas ke "vort" estas tabelo kies longo estas 8 kaj kies elementoj valoras unu el la videblaj karaktroj.

Ni deklaras "hodi" kun rikorda tipo "dat", tio signifas ke "hodi" estas rikordo de tri elementoj: "tag", "monat" kaj "jar".

Ni deklaras "pers" kun rikorda tipo "ul", tio signifas ke "pers" estas rikordo de ses elementoj: "nom", "anom", "nasktag", "seks", "famstat" kaj "profesi".

Ni deklaras "landf" kun ara tipo "flag", tio signifas ke "landf" estas aro de koloroj (iuj koloroj inter "blanka", "blua", "verda", "ruĝa", "nigra", "flava") aŭ malplena aro.

Ni deklaras "dentistar" kun rikordara tipo "ulrikar", tio signifas ke "dentistar" estas rikordaro kies elementoj estas rikordoj kun tipo "ul" kies elementoj estas "nom" kaj "numer".

Ni deklaras "libr" kun rikordara tipo "TEKST", tio signifas ke "libr" estas rikordaro de karaktraj elementoj ordigitaj laŭ linioj.

Ni deklaras "var0" kun ĉena tipo "nom"; tio signifas ke "var0" estas karaktra ĉeno kies longo estas maksimume 10.

Ni deklaras "k" kaj "l" kun montrila tipo "ref", tio signifas ke "k" kaj "l" estas montriloj al variabloj kun la tipo "par" kiu estas rikorda tipo kun du karaktraj elementoj "a" kaj "b".

III.2.5. Inito de variabloj.

Tiu parto povas ne ekzisti en programo.

En la ĉefa programo, la plej ekstera bloko, ni povas initi variablojn, tio estas : atribui komencan valoron al variablo. Tiun initon enkondukas la rezervita vorto VAL. Ne ĉiuj variabloj povas esti initataj en tiu parto.

Iuj kompileroj postulas ke en tiu parto, la variabloj aperu en la sama ordo en kiu ili aperis en la variablodeklara parto.

III.2.5.1. Inito de entjera, bulea, reela kaj karaktra variabloj.

Jen ekzemploj de validaj initoj de tiaj variabloj :

```
VAR a,b : ENT; c,d : REEL; e : BUL; f : KAR;
        % unue deklaro de la variabloj %
VAL a=2; b = -17; c = 0.3; d = 10.5E+4;
     e = VERA; f = 'B';
```

III.2.5.2. Inito de aliaj skalaraj variabloj.

Jen ekzemplo de inito de alia skalara variablo :

```
TIP kolor = (flava, verda, blua); % difino de tipo "kolor" %
VAR farb : kolor; % deklaro de variablo de la tipo "kolor" %
VAL farb = verda; % atribuo de valoro al la variablo "farb" %
```

III.2.5.3. Inito de intervala variablo.

Jen ekzemplo de valida inito de intervala variablo :

```
TIP ind_1 = 1..25; % deklaro de intervala tipo %
VAR i : ind_1; % deklaro de variablo de tipo "intervalo" %
VAL i = 5; % atribuo de valoro al la variablo %
```

III.2.5.4. Inito de tabela variablo.

Jen ekzemplo de valida inito de tabelaj variabloj :

```
TIP notb = TAB (.1..25.) DE ENT;
    damtab = TAB (.1..10,1..10.) DE BUL;
VAR valtab : notb; tabl,tab2 : damtab;
VAL valtab = 10,20,30,40,.......,250;
    tabl = FALSA, FALSA, FALSA, ......, FALSA;
```

III.2.5.5. Inito de rikorda variablo.

Jen ekzemplo de valida inito de rikorda variablo :

```
TIP blok = RIK a: REEL; jesne: BUL;
               i: ENT; k = KAR
           FIN; % deklaro de rikorda tipo %
VAR bl, b2 : blok; % deklaro de rikordaj variabloj %
VAL bl = (2.35, VERA, 12, 'C'); % atribuo de valoro %
    b2 = (3, FALSA, -627, '&');
```

III.2.5.6. Inito de ara variablo.

Jen ekzemplo de valida inito de ara variablo :

```
TIP kolor = (blanka, blua, verda, flava);
    flag = AR DE kolor;
VAR fl, f2 : flag;
VAL fl = (.blanka, verda.);
    f2 = (.blua, blanka, verda.);
```

III.2.5.7. Inito de ĉena variablo.

Jen ekzemplo de valida inito de ĉena variablo :

```
TIP vort = CHEN (.10.);
VAR verb, subst : vort;
VAL verb = 'fari'; subst = 'ni';
```

III.2.5.8. Inito de montrila variablo.

Ne eblas post la rezervita vorto "VAL".

III.2.5.9. Inito de rikordara variablo.

Ne eblas post la rezervita vorto "VAL" (?).

III.2.6. Deklaro de subrutinoj.

La kreo de subrutinoj celas eviti la ripetiĝon en pluraj lokoj de programo, de preskaŭ sama grupo de instrukcioj.

Estas pli bone kiam ni povas, logike partigi programon en moduleojn respondantajn al apartaj taskoj, eĉ se ni uzas ilin nur unufoje. Multaj estas la avantaĝoj de tia programado, la programo estas pli facile komprenebla, debugebla kaj modifebla, ĝia realigo povas pli facile partiĝi inter diversaj programistoj.

Plie en PASCAL-a lingvo, dank'al la blokstrukturo, eĉ se subrutino uziĝas nur unufoje, ni ŝparas storon, ĉar pluraj sinsekvaj subrutinoj (ne eningiĝaj) okupas la saman parton de storo por siaj apartaj variabloj, iliaj komunaj variabloj estas en pli vasta bloko kiu entenas la subrutinojn.

La deklaro de subrutino komenciĝas per la rezervita vorto "SUBRUTIN" sekva de identigilo kiu estas la nomo de la subrutino eventuale sekvata de listo de parametroj aŭ de la rezervita vorto "POSTE" se tiu subrutino estas difinita poste. Post tio sekvas bloko kiu povas enteni etikedodeklaran parton, konstantdifinan parton, tipdifinan parton, variablodeklaran parton, subrutindeklaran parton, funkcideklaran parton kaj instrukcitekstan parton sed ne variablinitan parton kiel povas enteni programbloko.

Jen ekzemploj de validaj subrutinaj deklaroj :

```
SUBRUTIN ord (d, e : ENT); POSTE;
SUBRUTIN sum (n : ENT; VAR k : ENT);
   EK
     ....          } % grupa instrukcifrazo %
   FIN;
SUBRUTIN integ (VAR i : REEL; a,b : REEL; FUNKCI f : REEL );
   EK
     ....          } % grupa instrukcifrazo %
   FIN;
```

La identigiloj deklaritaj en subrutino ne estas atingeblaj de ekster tiu subrutino. Tamen ĉiuj identigiloj deklaritaj en la blokoj kiuj entenas tiun subrutinon estas atingeblaj interne de tiu subrutino. Sed se la sama identigilo ekzistas en kaj ekster iu subrutino, nur la identigilo en la subrutino estas atingebla. Komence de ĉiu subrutino ĉiuj internaj variabloj (variabloj de la subrutino) enhavas nekonatan valoron.

Ni devas distingi la formalajn parametrojn kiuj estas uzataj en la deklaro de la subrutino por identigi kaj uzi la diversajn parametrojn en la subrutina bloko kaj la efektivajn parametrojn kiuj estas la realaj parametroj troviĝantaj dum la efektivigo de programo en efektiva voko al la subrutino.

Vi trovos en anekso 2 la liston de la jam difinitaj subrutinoj.

Subrutino povas enhavi referencon al si mem, tio estas kiam la identigilo de la subrutino "a" aperas en ĝia deklaro, tia subrutino estas nomata "rekte sinvoka subrutino". Se subrutino "a" enhavas en sia deklaro referencon al alia subrutino "b" kiu enhavas rektan aŭ nerektan referencon al la subrutino "a", tia subrutino "a" estas nomata "nerekte sinvoka subrutino".

La blokstrukturo de PASCAL ebligas al la programisto skribi sinvokajn subrutinojn sen pli atenti pri la apartecoj de tiaj subrutinoj.

Kaze de nerekte sinvoka subrutino, la voko de subrutino povas antaŭi ĝian difinon. La filozofio de PASCAL malpermesas tion, sed ni solvas tiun problemon, anoncante ĝin kun la rezervita vorto "POSTE" anstataŭ la bloko de la subrutino.

Jen ekzempleroj de tiaj validaj subrutinoj :

```
SUBRUTIN a ( i : ENT ); POSTE;
SUBRUTIN b(j:ENT);
   EK
      ....; a(e);.....        % grupa instrukcifrazo %
   FIN;
SUBRUTIN a;
   EK
      ....; b(f); ...         % grupa instrukcifrazo %
   FIN;
....
```

Ni rimarku ke kiam ni kompletigas la difinon de la subrutino "a", ni ne plu indikas la parametrojn.

La parametroj de subrutino povas esti "valoro", "variablo", "subrutino" aŭ "funkcio".

III.2.6.1. Parametro: "valoro".

En antaŭa ekzemplo, "n" en la subrutino "sum" kaj "a" kaj "b" en la subrutino "integ" estas parametroj-valoroj.

Neniu rezervita vorto antaŭas tian parametron. La efektiva parametro devas esti ekspresio (variablo estas simpla kazo de ekspresio). La formala parametro estas loka variablo por la vokita subrutino kiu ricevas ĉe la voko por inita valoro la valoron de la efektiva parametro. En la subrutino, ni povas modifi tiun formalan parametron sed ĝia valoro restos interne de la subrutino. Se la efektiva parametro povas esti granda struktura variablo, por eviti ĝian rekopion ĉe la voko, pli bone estas uzi parametron-variablon.

III.2.6.2. Parametro: "variablo".

En antaŭa ekzemplo, "k" en la subrutino "sum" kaj "i" en la subrutino "integ" estas parametroj-variabloj.

La rezervita vorto "VAR" antaŭas tian parametron. La efektiva parametro devas esti variablo; ĉe la voko, adreso proviziĝas kiel parametro. La adreskalkulo okazas nur unufoje ĉe la voko, tiel se "a(.i.)" estas efektiva parametro, poste en la subrutino ni povas modifi la variablon "i" sed la parametro ĉiam havas la adreson kiun ĝi havis je la vokmomento.

Ni devas uzi parametron-variablon kiam ni volas doni rezulton al la vokinta programo.

III.2.6.3. Parametro: "subrutino".

La rezervita vorto "SUBRUTIN" antaŭas tian parametron. La subrutinoj uzataj kiel efektiva parametro devas havi nur parametrojn-valorojn.

III.2.6.4. Parametro: "funkcio".

En antaŭa ekzemplo "f" en la subrutino "integ" estas parametro-funkcio.

La rezervita vorto "FUNKCI" antaŭas tian parametron kaj indiko de ĝia tipo devas posti ĝin.

Funkcio uzata kiel efektiva parametro devas havi nur parametrojn-valorojn.

III.2.7. Deklaro de funkcioj.

Funkcio similas al subrutino sed provizas al la vokinta programo rezulton de tipo : skalara aŭ montrila kiu povas esti uzata en ekspresio pro tio.

Jen ekzemplo de valida funkcia deklaro :

```
FUNKCI eksponen (x: REEL, y: ENT) : REEL;
  VAR z: REEL;
  EK                     )
    ...; eksponen := z   } % grupa instrukcifrazo %
  FIN;                   )
  .....
```

En anekso 3 troviĝas listo de la jam difinitaj funkcioj.

III.3. Instrukcifrazoj.

En multaj instrukcioj troviĝas ekspresio pli aŭ malpli simpla konsistanta el operandoj kaj operatoroj.

La operandoj estas deklaritaj identigiloj (variabloj, konstantoj, funkcioj) aŭ konstantoj.

Jen la operatoroj en grupoj kun malkreska prioritato :

operatoro	operacio	operandtipoj	rezultotipo
1: NE	negacio	bulea	bulea
2: *	multipliko ara kruciĝo	entjeraj, reelaj iuj ajn aroj de tipo T	entjera, reela tipo T

DIV	divido kun entjerigo	entjeraj	entjera
/	divido	entjeraj, reelaj	entjera, reela
MOD	moduluso	entjera	entjera
KAJ	logika "KAJ"	buleaj	bulea
3: +	adicio ara kuniĝo	entjeraj, reelaj iuj ajn aroj de tipo T	entjera, reela tipo T
-	subtraho ara diferenco	entjeraj, reelaj iuj ajn aroj de tipo T	entjera, reela tipo T
AU	logika "AU"	buleaj	bulea

Operatoro	operacio	operandtipoj	rezultotipoj
4: =	"egal"	Iuj ajn skalaraj, intervalaj aŭ araj tipoj	bulea
<>	"neegal"	Iuj ajn skalaraj, intervalaj aŭ araj tipoj	bulea
<	"sub"	Iuj ajn skalaraj, intervalaj tipoj (ne araj)	bulea
>	"super"	Iuj ajn skalaraj, intervalaj tipoj (ne araj)	bulea
<=	"subegal" aŭ "enestas"	Iuj ajn skalaraj, intervalaj aŭ araj tipoj	bulea
>=	"superegal" aŭ "enhavas"	Iuj ajn skalaraj, intervalaj aŭ araj tipoj	bulea
EN	"apartenas al"	Iu ajn skalara, intervala tipo kaj ĝia ara tipo	bulea

En tiuj tabeloj, ni trovas la operatorojn pri la aroj :

* (ara kruciĝo) : a * b (ĉiuj elementoj samtempe en "a" kaj en "b").

+ (ara kuniĝo) : a + b (ĉiuj elementoj aŭ en "a" aŭ en "b").

- (ara diferenco) : a - b (ĉiuj elementoj en "a" kiuj ne estas en "b").

EN (aparteno) : a EN b (VERA se la elemento "a" apartenas al la aro "b").

Kiam la operatoroj estas samprioritataj, la operacioj estas efektivigataj de maldekstre dekstren. Kompreneble ekspresio inter krampoj estas efektivigata unue sendepende de la eksteraj operatoroj.

III.3.1. La baza instrukcifrazo : la atribuo.

Modelo :   .... := ....

Tiu instrukcifrazo signifas ke la valoro kalkulita dekstre de la atribua operatoro ": " devas esti atribuata al la variablo ĉe la maldekstra flanko de la atribua operatoro.

Jen ekzemploj de validaj atribuaj instrukcifrazoj :

```
i := 5; % atribuo de 5 al entjera, intervala aŭ reela variablo %
r := 2.3; % atribuo de 2.3 al reela variablo %
k := 'C'; % atribuo de 'C' al karaktra aŭ ĉena variablo %
b := VERA; % atribuo de "VERA" al bulea variablo %
c := FALSA; % atribuo de "FALSA" al bulea variablo %
a(.3.) := 4; % atribuo de 4 al la 3a elemento de tabela variablo %
alan.nasktag.monat := 9; % atribuo de 9 al la elemento "monat" de
                       rikordo "nasktag" de rikordo "alan" %
miks := (.flava, verda.); % atribuo de "flava" kaj "verda" al
                       ara variablo %
po@.a := 5; % atribuo de 5 al elemento "a" de rikordo montrata
                       de la montrila variablo "po" %
```

III.3.2. La subrutinaj instrukcioj.

Ili estas necesaj por efektivigi subrutinon.

Jen ekzemploj de validaj subrutinvokoj :

```
prez(refl,kvl); % voko de subrutino "prez" kun du parametroj
                "refl" kaj "kvl" %
sekv; % voko de subrutino "sekv" sen parametro %
```

Por pliaj detaloj pri la parametroj, vidi la sekcion pri la difino de la subrutinoj.

III.3.3. La instrukcifrazo IRIAL.

Modelo : IRIAL ......

Instrukcifrazo IRIAL utilas por indiki ke la efektivigo de la instrukcioj ne plu laŭsekvence devas daŭri sed redaŭri en alia parto de la programo ĝuste kie troviĝas la etikedo indikita en la instrukcifrazo IRIAL. Ne forgesu ke tiu etikedo devas esti deklarita en aparta sekcio post la rezervita vorto "ETIK". Plie ne eblas salti en subrutinon aŭ el subrutino.

Ni forte rekomendas kiel eble malplej enkonduki etikedojn en programon.

Jen ekzemploj de instrukcifrazoj IRIAL :

```
...
ETIK 12,stop;
 ...
IRIAL 12;
 ...
IRIAL stop;
```

III.3.4. La malplena instrukcio.

Indikas ĝin neniu simbolo kaj neniu ago rilatas al ĝi. Tiu instrukcio aperas kiam la sintakso de la lingvo PASKAL postulas instrukcion kaj kiam neniu instrukcio ekzistas.

Jen ekzemplo :

```
EK
FIN;
```

III.3.5. La elira instrukcifrazo ELIR.

Modelo : ELIR;

La efektivigo de elira instrukcifrazo sen etikeda operando estigas daŭron de la programa efektivigo tuj post la plej malgranda maŝa instrukcifrazo (POR, DUM, RIPETI) kiu enhavas ĝin.

Por ekzemplo :

```
EK
  ...
  POR
    ...
    ...; ELIR;
  FIN;
  % la programefektivigo redaŭras ĉi tie post efektivigo de la
    instrukcifrazo ELIR %;
FIN;
```

La efektivigo de elira instrukcio kun etikeda operando estigas daŭron de la programa efektivigo tuj post la maŝa instrukcifrazo (POR, DUM, RIPETI) kiu surhavas tiun etikedon.

Por ekzemplo :

```
etl: DUM
       ...
         POR .....
         RIPETI ...
           ...; ELIR etl;
         FIN;
         ...
     FIN;
     % la programefektivigo redaŭras ĉi tie post efektivigo de la
       instrukcifrazo ELIR etl %;
```

III.3.6. La grupa instrukcifrazo.

Modelo : EK .... FIN;

La grupa instrukcifrazo estas sekvenco de instrukcifrazoj kiuj devas esti efektivigataj en la ordo en kiu ili aperas. La rezervitaj vortoj "EK" kaj "FIN" limigas tiun grupan instrukcifrazon.

Jen ekzemplo :

```
EK z := a;
   a := b;
   b := z;
FIN; %interŝanĝo de "a" kaj "b" tra "z" %;
```

III.3.7. La instrukcifrazo KUN.

Modelo : KUN ...... FARI ....;

En tiu instrukcifrazo, la elementoj de rikorda(j) variablo(j) povas esti pli simple atingeblaj por la programisto per indiko de malpli granda kontekso.

Por ekzemplo :

```
TIP dat = RIK tag, monat, jar : ENT
          FIN;
    ul = RIK nom, anom : CHEN (.10.);
              nasktag : dat;
VAR alan : ul;
% sen instrukcifrazo KUN : %
    alan.nom := 'rapid';
    alan.anom := 'tre';
    alan.nasktag.tag := 12;
    alan.nasktag.monat := 5;
    alan.nasktag.jar := 1963;
% kun instrukcifrazo KUN : %
    KUN alan, nasktag FARI
    EK
       nom := 'rapid'; anom := 'tre';
       tag := 12; monat := 5; jar := 1963
    FIN;
```

III.3.8. La kondiĉa instrukcifrazo SE.

Modelo : SE ..... DO ...... [SENE ......;]

Instrukcifrazo SE indikas ke la instrukcifrazo troviĝanta post "DO" devas esti efektivigata nur se aparta kondiĉo (bulea ekspresio) estas "VERA". Se tiu kondiĉo estas "FALSA", aŭ neniu instrukcifrazo estas efektivigota aŭ se la rezervita vorto "SENE" enestas, la instrukcifrazo kiu sekvas tiun vorton "SENE" estas efektivigota.

Por ekzemplo :

```
SE monat = 12 DO
   EK monat := 1; jar := jar+1
   FIN
SENE monat := monat+1;
```

Atentu : neniu punktokomo devas troviĝi antaŭ la rezervita vorto "SENE".

En kazo de sinsekvaj instrukcifrazoj SE, ĉiu "SENE" rilatas al la plej proksima antaŭa "SE".

Tial : SE ..... DO SE ...... DO .....

SENE ......

samas al : SE .... DO

```
EK
   SE ...... DO ........
   SENE ......
FIN
```

III.3.9. La kondiĉa instrukcifrazo KAZ.

Modelo : KAZ ...... INTER ...... [ALIE ......] FIN;

La instrukcifrazo KAZ konsistas el ekspresio (selektilo) kaj el listo de instrukcifrazoj, ĉiu el ili kun etikedo kies tipo samas al la tipo de la selektilo. Nur la instrukcifrazo kies etikedo egalas la valoron de la selektilo estos efektivigata. Se neniu etikedo egalas la valoron de la selektilo, la instrukcifrazo post la rezervita vorto "ALIE" se ĝi enestas estos efektivigata.

Por ekzemplo :

```
KAZ operator INTER
    plus: a := a+b;
    minus: a := a-b
FIN;
KAZ stat INTER
    preta: EK i:=i+1; stat := aktiva FIN;
    aktiva:; % ni faras nenion en tiu stato %
    okupata, nefunkcia: i:=i+2
    ALIE: i:=0
FIN;
```

III.3.10. La maŝa instrukcifrazo DUM.

Modelo : DUM ...... FARI ....;

La maŝa instrukcifrazo **DUM** konsistas el bulea ekspresio kaj instrukcifrazo. Tiu instrukcifrazo estas ripete efektivigata dum la rega bulea ekspresio estas "VERA". Se la bulea ekspresio estas "FALSA" ĉe la komenco, la instrukcifrazo tute ne estas efektivigata.

Por ekzemplo :

```
DUM a<3 FARI a:= a+b;
% Tiu instrukcifrazo DUM ekvivalentas al :%
   kom: SE a<3 DO
          EK
             a:= a+b; IRIAL kom
          FIN;
```

Jen alia ekzemplo : serĉo de elemento en tabelo "a" ekde a(.0.) :

```
i:= 0;
DUM a(.i.) < elem FARI
     SE i=n DO ELIR; % "elem" ne trovita %
     i:=i+1;
```

III.3.11. La maŝa instrukcifrazo RIPETI.

Modelo : RIPETI .... GHIS ....;

La maŝa instrukcifrazo RIPETI konsistas el instrukcifrazo kaj el bulea ekspresio. La instrukcifrazo estas almenaŭ unufoje efektivigata kaj ripete efektivigata se la bulea ekspresio estas "FALSA". Kiam la bulea ekspresio fariĝas "VERA", la ripeta efektivigo de la instrukcifrazo finiĝas.

Tiu instrukcifrazo : RIPETI %instrukcifrazo% GHIS %bulea ekspresio% ekvivalentas al :

```
etik: %instrukcifrazo%;
     SE NE %bulea ekspresio% DO IRIAL etik;
```

Por ekzemplo : serĉo de elemento en tabelo "a" :

```
% ni serĉas la elementon "elem" ekde a(.1.) %
i:= 0;
RIPETI i:=i+1;
       SE i=n+1 DO ELIR
GHIS a(.i.) = elem;
```

III.3.12. La maŝa instrukcifrazo POR.

Modelo 1 : POR ...... AL ...... FARI ....;

Modelo 2 : POR ......SUBAL .... FARI ....;

La maŝa instrukcifrazo POR indikas ke instrukcio devas esti ripete efektivigata dum progresio de valoroj atribuitaj al variablo nomata la "regvariablo" de la instrukcifrazo. La progresio povas supreniri aŭ malsupreniri al fina valoro.

La regvariablo, la komenca valoro kaj la fina valoro devas havi skalaran tipon (aŭ intervalan por la regvariablo) kaj ne devas esti modifataj en la ripetata instrukcifrazo.

La instrukcifrazo POR i:=b1 AL b2 FARI %instrukcifrazo%; ekvivalentas al :

```
EK
   i:= b1;
   SE i <=b2 DO
    RIPETI
      %instrukcifrazo %
      i:=POST(i);
    GHIS i > b2
FIN
```

La instrukcifrazo POR i:=d1 SUBAL d2 FARI %instrukcifrazo%; ekvivalentas al :

```
EK
   i:=d1;
   SE i>=d2 DO
    RIPETI
      %instrukcifrazo %
      i:=ANT(i);
    GHIS i < d2
FIN
```

Por ekzemplo, sumo de la elementoj de tabelo :

```
sum := 0; n := 5; m :=10;
POR i:=1 AL n FARI
    POR j:=1 AL m FARI
       sum := sum+a(.i,j.);
```

Jen alia ekzemplo :

```
TIP tag = (lun, mar, mer, jaŭ, ven, sab, dim);
VAR t:tag; labhor: TAB (.tag.) DE ENT;
    horsum: ENT; % sumo de la laborhoroj %
VAL horsum=0;
EK
  POR t:=lun AL ven FARI
       horsum := horsum+labhor(.t.)
FIN
```

III.4. Operacioj pri la diverstipaj variabloj.

Unue kelkaj funkcioj komunaj al pluraj variablotipoj.

Jen ekzemplo de deklaroj por klarigi la funkciojn :

```
TIP kolor = (blua,flava,verda,nigra,blanka);
TIP valut = (dolar,pund,frank,mark,gulden);
VAR v:valut; k:kolor;
```

Jen funkcioj pri tiuj variabloj :

Funkcio ANT : ANT(v) liveras la valoron kiu antaŭas la valoron de "v" se ĝi ekzistas, aŭ nekonatan valoron, se ne ekzistas antaŭa valoro. Tiu funkcio ne povas esti aplikata al reela variablo.

Por ekzemplo :

```
ANT(frank) liveras "pund"
ANT(blanka) liveras "nigra"
ANT(6) liveras "5"   por entjera variablo
```

Kun karaktra variablo "liter", ANT(liter) liveras la karaktron kiu antaŭas la karaktron indikatan de la variablo "liter" (vidi anekson 4).

Funkcio POST : POST(v) liveras la valoron kiu postas la valoron de "v" se ĝi ekzistas, aŭ nekonatan valoron, se ne ekzistas posta valoro. Tiu funkcio ne povas esti aplikata al reela variablo.

Por ekzemplo :

POST(blua) liveras "flava"
POST(mark) liveras "gulden"
POST('D') liveras 'E'
POST(12) liveras "13" por entjera variablo

Funkcio RANG : RANG(v) liveras la rangon de la valoro de "v" en la aro de la elementoj de la tipo de "v". La unua elemento havas la rangon "0" (nulo).

Por ekzemplo :

RANG(dolar) liveras "0"
RANG(verda) liveras "2"
RANG('C') liveras "3" (vidi anekson 4)

Funkcio KR : KR(v) kie "v" estas entjera variablo liveras la karaktron de rango indikata per la enhavo de la variablo "v".

Por ekzemplo :

KR(1) liveras "A" (vidi anekson 4)

En la sekvantaj ekvivalentaj ekspresioj, "k" estas karaktra variablo :

POST (k) ekvivalentas al KR (RANG(k) + 1)
ANT (k) ekvivalentas al KR (RANG(k) - 1)

III.4.1. apartaj operacioj pri la skalaroj.

La funkcioj ANT, POST kaj RANG aplikiĝas al ili.
La rilataj operatoroj "OP" aplikiĝas al ili tiel :
v1 OP v2   se   RANG(v1) OP RANG(v2)
kie OP estas unu el la sekvantaj simboloj : = < <= >= > <>.

III.4.2. apartaj operacioj pri la buleaĵoj.

Inter buleaj variabloj aplikiĝas la operatoroj "KAJ", "AU" kaj "NE".
Bulean valoron liveras ĉiuj rilataj operatoroj = < <= >= > <> kaj "EN".

Pro la fakto ke la bulea tipo difiniĝis kiel TIP BUL = (FALSA,VERA) la rilataj operatoroj aplikiĝas al ili kaj same la funkcioj ANT, POST kaj RANG.

III.4.3. apartaj operacioj pri la entjeroj.

La sekvantaj operatoroj pri entjeroj liveras entjeron :

* : multipliko
DIV : entjera divido
MOD : moduluso : A MOD B = A - ((A DIV B) * B)
+ : adicio
- : subtraho

La rilataj operatoroj "OP" = < <= >= > <> aplikiĝas al entjeroj kaj liveras bulean rezulton :

e1 OP e2 se RANG(e1) OP RANG(e2)

Jen la jam difinitaj funkcioj kiu liveras entjeran rezulton :

ABS (z) : absoluta valoro de la entjero "z"
KV (z) : kvadrato de la entjero "z" (aŭ "z" je potenco 2)
TRUNK (z) : entjera parto de la reelo "z"

Por ekzemplo :

18 DIV 4 liveras "4"
18 MOD 4 liveras "2"
KV (5) liveras "25"
TRUNK (3.15) liveras "3"
ABS (-35) liveras "35"

Ankaŭ la funkcioj ANT, POST kaj KR aplikiĝas al entjero. La funkcio KR aplikiĝas al entjero nur de "0" ĝis "63".

III.4.4. apartaj operacioj pri la reeloj.

La sekvantaj operatoroj pri reeloj liveras reelon :

* : multipliko
/ : divido (divido de du entjeroj tamen liveras reelon)
+ : adicio
- : subtraho

La sekvantaj funkcioj aplikiĝas al reelo :

ABS (z) : absoluta valoro de reelo "z"
KV (z) : kvadrato de reelo "z"
SIN (z) : sinuso de reelo "z" (angulo en arkuso)
KOS (z) : kosinuso de reelo "z" (angulo en arkuso)
ATAN (z) : arktangento de reelo "z"

NL (z) : nepera logaritmo de reelo "z"
EKSP (z) : e^z : eksponencialo de reelo "z"
KVR (z) : $\sqrt{z}$: kvadrata radiko de reelo "z"

Reela tipo iom malsamas al la aliaj skalaraj tipoj, ni ne povas difini nek antaŭan nek postan reelon de reelo, ni ne povas uzi ĝin nek kiel indicon nek kiel bazon de aro, nek kiel maŝregan variablon.

III.4.5. apartaj operacioj pri la karaktroj.

La funkcioj RANG kaj KR ebligas rilatigi la entjerojn de "0" ĝis "63" al la karaktroj (vidi anekson 4).

KR (p) liveras la karaktron kies rango estas la valoro de "p".

RANG (k) liveras la rangon de la karaktro de "k".

Rimarku ke RANG(KR(p)) = p kaj KR(RANG(k)) = k

La rilataj operatoroj "OP" = < <= >= > <> aplikiĝas al karaktroj kaj liveras bulean rezulton :

k1 OP k2 se RANG(k1) OP RANG(k2)

Ankaŭ la funkcioj ANT kaj POST aplikiĝas al karaktroj.

III.4.6. apartaj operacioj pri la intervaloj.

La operatoroj kaj la funkcioj kiuj aplikiĝas al la skalaroj aplikiĝas ankaŭ al la intervaloj.

III.4.7. apartaj operacioj pri la tabeloj.

La operacioj kiuj aplikiĝas al tabelo estas la operacioj kiuj aplikiĝas al ĝiaj elementoj depende de iliaj tipoj : entjero, reelo, buleaĵo, karaktro.

Ni atingas elementon de tabelo per indiko de la tabelnomo kaj eventuale de indico(j) : t(.i,j,k.) aŭ t. En tiu ekzemplo la tabelo "t" estas tri-dimensia.

Por ekzemplo :

```
VAR t1, t2 : TAB (.1..10.)DE ENT;
    t3 : TAB (.1..10,1..10.) DE ENT;
    t1 := t2;
    t1 (.i.) := t2 (.i.) + t1 (.i.);
    t3 (.5.) := t2;
```

Kiam ni havas kompaktajn tabelojn kaj ofte necesas atingi iliajn elementojn, eblas malkompaktigi ilin kaj rekompaktigi ilin per uzo de du subrutinoj : MKOMP kaj KOMP.

Se "t" estas KOMPA TAB (.u..v.) DE ENT kaj "a" estas TAB (.m..n.) DE ENT kun (n-m)   (v-u) tiam :

KOMP(a,i,t) liveras en la tabelo "t" kompaktigitan tabelon el la tabelo "a".

KOMP (a,i,t) ekvivalentas al :

```
POR j := u AL v FARI t(.j.) := a(.j-u+i.);
```

MKOMP(t,a,i) liveras en la tabelo "a" malkompaktigitan tabelon el la tabelo "t".

MKOMP(t,a,i) ekvivalentas al :

```
POR j := u AL v FARI a(.j-u+i.) := t(.j.);
```

III.4.8. apartaj operacioj pri la rikordoj.

La operacioj kiuj aplikiĝas al rikordo estas la operacioj kiuj aplikiĝas al ĝiaj elementoj depende de iliaj tipoj : entjero, reelo, buleaĵo aŭ karaktro.

Ni atingas elementon de rikordo per indiko de la rikorda variablo kaj de iu kampnomo.

Por ekzemplo :

```
TIP dat = RIK tag : 1..31;
              monat : 1..12;
              jar : 1900..2100
          FIN;
    adr = RIK strat : CHEN (.20.);
              urbkod : CHEN (.6.);
              urb : ALFA
          FIN;
```

```
    ul = RIK nom, anom : ALFA;
             nasktag : dat;
             ladr : adr      % loĝadreso %
         FIN
VAR abonant : ul;
    abonant.nom := 'TRAJNO';
    abonant.anom := 'rapida';
    abonant.nasktag.tag := 1;
    abonant.nasktag.monat := 06;
    abonant.nasktag.jar := 1982;
    abonant.ladr.strat := '2, strato de la stacidomo';
    abonant.ladr.urbkod := '38145K';
    abonant.ladr.urb := 'ANASKORT';
```

Pli simplan atingon de la elementoj ebligas la instrukcio KUN.

III.4.9. apartaj operacioj pri la aroj.

Jen la uzeblaj operatoroj pri la aroj :

+ : unuiĝo : A+B estas la aro de la elementoj kiuj apartenas aŭ al A aŭ al B aŭ al ambaŭ.

* : kruciĝo: A*B estas la aro de la elementoj kiuj apartenas al A kaj ankaŭ al B.

\- diferenco : A-B estas la aro de la elementoj kiuj apartenas al A kaj ne al B.

EN aparteno : A EN B liveras la bulean rezulton VERA se la elemento aŭ la intervalo A estas en B, FALSA se la elemento aŭ la intervalo A ne estas en B.

= egaleco : A=B liveras la bulean rezulton VERA se A egalas B (FALSA alikaze).

<> malegaleco : A <> B liveras la bulean rezulton VERA se A ne egalas B (FALSA alikaze).

<= enesto : A<=B liveras la bulean rezulton VERA se A enestas en B (FALSA alikaze).

>= enhavo : A>=B liveras la bulean rezulton VERA se A enhavas B (FALSA alikaze).

Por ekzemplo :

```
TIP kolor = (blanka, nigra, blua, verda, flava);
VAL miks, flag : kolor;
VAL miks := (..);      % malplena aro %
    flag := (.blanka,nigra.);
    miks := (.flava.);
    SE verda EN flag DO ......
```

III.4.10. apartaj operacioj pri la netekstaj rikordaroj.

La deklaro de variablo "r" kun rikordara tipo aŭtomate kreas "fenestron" atingeblan per r∂ kun la tipo de la elementoj de la rikordaro "r". Tio signifas ke je iu ajn momento nur unu elemento de la rikordaro estas atingebla pere de r∂.

Por ekzemplo :

```
TIP pers = RIK nom,anom : CHEN (.10.);
               numer : ENT; nots : 1..10   % notsumo %
           FIN;
    rpers = RIKAR DE pers;
    rnot = RIKAR DE 1..10;
VAR rlernant : rpers;
    rnmat, rnfiz : rnot; %rikordaroj de notoj pri matematiko %
    rnlit, rnhis : rnot; %pri fiziko, literaturo, historio %
    rlernanto.nots := rnmat∂ + rnfiz∂ + rnlit∂ + rnhis∂;
      % en la lernanta rikordaro, ni metas la sumon de la notoj %
```

Pere de tiu fenestro r∂, ni povas atingi ĉiujn elementojn de la rikordaro kaj plilongigi ĝin per skribo de pliaj elementoj. Pro la fakto ke nur unu elemento de la rikordaro estas atingebla je iu ajn momento, ni disponas pri subrutinoj aŭ funkcioj por movi tiun fenestron kaj tiel atingi ĉiujn elementojn de la rikordaro.

Funkcio FDR (Fino De Rikordaro) : kiam la fenestro r∂ troviĝas ĉe la fino de la rikordaro "r" (t.e. post la lasta elemento), FDR(r) = VERA, alikaze FDR(r) = FALSA.

Subrutino LRES (LegREStarto) : LRES(r) movas la fenestron r@ fronte al la unua elemento de la rikordaro, t.e. per r@ ni atingas la unuan elementon de la rikordaro. Se la rikordaro enhavas elementojn, tiam FDR(r) = FALSA. Se la rikordaro estas malplena, tiam FDR(r) = VERA, kaj la enhavo de r@ estas nekonata. LRES(r) estas deviga antaŭ la unua rikordara lego.

Subrutino SRES (SkribREStarto) : SRES(r) malplenigas la rikordaron "r", FDR(r) fariĝas VERA, post tio la unua elemento povas esti enigata. SRES(r) estas deviga antaŭ la unua rikordara skribo.

Subrutino AKIR (AKIRo de rikordo) : AKIR(r) modifas la pozicion de la fenestro r@ tiel ke ĝi frontas la sekvantan elementon, se ĝi ne ekzistas FDR(r) fariĝas VERA kaj la enhavo de r@ ne estas konata. Fatala eraro okazas se ni provas voki AKIR(r) kiam FDR(r) jam estas VERA.

Subrutino MET (METo de rikordo) : MET(r) aldonas ĉe la fino de la rikordaro "r" la nunan enhavon de la fenestro r@. Tio eblas nur se FDR(r) = VERA, t.e. se ni frontas la finon de la rikordaro.

De la priskribo de tiuj subrutinoj kaj funkcio ni konstatas ke ni ne povas modifi la elementojn de rikordaro, ni povas nur krei novan rikordaron el la malnova modifante aŭ aldonante iujn elementojn.

Por ekzemplo :

Du rikordarojn de ordaj entjeroj ni devas kunigi en unu rikordaron de ordaj entjeroj.

```
PROG kunordig; % kunordigo de du rikordaroj de ordaj entjeroj
                 "a" kaj "b" en unu rikordaron "c" %
VAR a, b, c : RIKAR DE ENT;
EK
  LRES(a); LRES(b); SRES(c);
  DUM NE FDR(a) KAJ NE FDR(b) FARI
  EK
    SE a@ < b@ DO
    EK c@ := a@; AKIR(a)
    FIN
    SENE
    EK c@ := b@; AKIR(b)
    FIN
    MET(c)
```

```
   FIN;
   % unu el la rikordaroj "a" aŭ "b" estas legita %
   DUM NE FDR(a) FARI % ni kopias la finon de "a" %
   EK
     c@ := a@; AKIR(a); MET(c)
   FIN;
   DUM NE FDR(b) FARI % ni kopias la finon de "b" %
   EK
     c@ := b@; AKIR(b); MET(c)
   FIN
  FIN.
```

III.4.11. apartaj operacioj pri la tekstaj rikordaroj.

La tekstaj rikordaroj enhavas elementojn nur kun la karaktra tipo (KAR).

Apartigas ilin de la normalaj rikordaroj la partiĝo en linioj kies fino indikas neenesta karaktro generata kaj detektebla per apartaj operatoroj.

Jen la funkcio kaj la subrutinoj kiuj koncernas ilin :

Funkcio FDL : kiam la fino de la linio estas atingita FDL (tr) = VERA kaj tr@ frontas spacon kaj ne la linifinan karaktron, alikaze FDL(tr) = FALSA.

Subrutino LFSKRIB : LiniFina SKRIBo. LFSKRIB (tr) finas la nunan linion lokigante la linifinan karaktron. Pliaj detaloj post la priskribo de la subrutino SKRIB.

Subrutino LEGNL : LEGo post iro komence de Nova Linio. Se restas karaktroj en la nuna linio LEGNL(tr) ignoras ilin. Post LEGNL(tr) la fenestro tr@ frontas la unuan karaktron de la posta linio kaj FDR(tr) = FALSA se posta linio ekzistas. Se posta linio ne ekzistas tiam FDR(tr) = VERA kaj tr@ ne estas konata same kiel por la aliaj rikordaroj.

Subrutino PAGH : poziciigo ĉe la komenco de la sekvanta paĝo en eliga rikordaro "tr". Pliaj detaloj post la subrutino SKRIB.

Por ekzemplo :

```
PROG tekstkopi; %kopio de tekstrikordaro %
VAR fr,cr : TEKST; %fonta kaj cela rikordaroj %
    k : KAR;
EK LRES(fr); SRES(cr);
  DUM NE FDL(fr) FARI
  EK %kopio de unu linio %
   DUM NE FDL(fr) FARI
    EK
      cr@ := fr@; AKIR(fr); MET(cr)
    FIN
  FIN
  LEGNL(fr); LFSKRIB(cr)
FIN.
```

La jam difinitaj tekstaj rikordaroj : ENIGO kaj ELIGO.

La periferiaparatoj estas konsiderataj kiel apartaj tekstaj rikordaroj. Ni vidis subrutinojn por manipuli tiujn rikordarojn MET, AKIR kiu ĉiufoje agas al unu elemento de la rikordaro (por la teksta rikordaro, karaktro estas elemento), ĉar estus malfacile kaj longe konverti reelon (por ekzemplo) en karaktran vicon por presi ĝin, ekzistas jam difinitaj subrutinoj LEG kaj SKRIB (poste priskribitaj) kiuj konvertas la datenojn de interna kodo al karaktra vico kaj inverse.

Aliflanke, du tekstaj rikordaroj estas jam difinitaj : ENI O por enigi la datenojn, kutime ligita al la kartlegilo kaj ELIGO por eligi la datenojn, kutime ligita al la printero. Komence de ĉiu PASCALa programo, ne necesas fari LRES(ENIGO) kaj SRES(ELIGO) kaj eĉ poste tion oni ne devas fari.

La efektivaj enigoj aŭ eligoj ne okazas samtempe kun la ordonoj en la programo, ĉar por ĉiu teksta rikordaro, la sistemo administras bufron kiu povas enteni tutan linion.

Por lego, tiu bufro pleniĝas per la unua ordono (LEG aŭ LEGNL) aŭ per la uzo de r@, FDR(r) aŭ FDL(r). Poste la sekvantaj ordonoj rilatas al la enhavo de tiu bufro ĝis ellego aŭ ĝis kiam la programo ordonas legon de nova linio.

Por skribo, la sinsekvaj skribaj ordonoj plenigas la bufron ĝis kiam la programo ordonas eligon al la periferiaparato.

Jen la subrutinoj LEG kaj SKRIB :

Subrutino LEG : tiu subrutino ne similas al la aliaj PASCALaj subrutinoj ĉar ĝi povas havi diverstipajn kaj diversnombrajn parametrojn. LEG(r) uziĝas por akiri elemento(j)n el la rikordaro "r". Jen variabloj v1, v2, ...vn kun karaktra tipo (KAR) aŭ entjera tipo (ENT) aŭ reela tipo (REEL) kaj la teksta rikordaro tr (TEKST).

Por ekzemplo :

```
LEG (tr,v1) : por legi la variablon v1 el la bufro de "tr"
LEG (v1)    : ekvivalentas al LEG(ENIGO,v1)
LEG (tr,v1,v2,...vn) ekvivalentas al :
            EK LEG(tr,v1); LEG(tr,v2); ....LEG(tr,vn) FIN
LEG (v1,v2,...vn) ekvivalentas al :
            EK LEG(v1); LEG(v2); .....LEG(vn) FIN
              aŭ ekvivalentas al :
            EK LEG(ENIGO,v1); ....LEG(ENIGO,vn) FIN
LEGNL(tr) ignoras la finon de la bufro kaj legas la novan
          linion de la rikordaro "tr".
LEGNL ekvivalentas al LEGNL (ENIGO).
LEGNL (tr,v1,v2,...vn) ekvivalentas al :
          EK LEG(tr,v1); .....LEG(tr,vn); LEGNL(tr) FIN
LEGNL (v1,v2,...vn) ekvivalentas al :
          EK LEG(v1); .....LEG(vn); LEGNL FIN
                aŭ ekvivalentas al :
          EK LEG(ENIGO,v1); ....LEG(ENIGO,vn); LEGNL(ENIGO) FIN
```

La valoroj de v1, v2, ...vn povas troviĝi sur pluraj linioj, la sistemo legos la necesajn liniojn por plenumi la leginstrukcion.

Ne necesas komenci per LEGNL ĉar la sistemo ne havas jam okupitan bufron kaj pro tio la unua operacio pri la rikordaro ENIGO estigos la plenigon de la bufro.

La legataj nombroj devas konformi la sintakson de PASCAL kaj esti apartigitaj per almenaŭ unu spaco aŭ linifino.

Subrutino SKRIB : (skribo de unu aŭ pluraj elementoj).

Jen la parametroj p1,p2,...pn kaj la teksta rikordaro "tr".

Por ekzemplo :

```
SKRIB(tr,p1) lokigas la valoron de p1 post konverto en la
             bufron de "tr".
SKRIB(p1) ekvivalentas al SKRIB (ELIGO,p1)
```

```
SKRIB (tr,pl,...pn) ekvivalentas al :
      EK SKRIB(tr,pl); .....SKRIB(tr,pn) FIN
SKRIB (pl,....pn) ekvivalentas al :
      EK SKRIB (pl); ......SKRIB(pn) FIN
               aŭ ekvivalentas al :
      EK SKRIB(ELIGO,pl); ....SKRIB(ELIGO,pn) FIN
LFSKRIB (tr) skribas linifinon kaj malplenigas la bufron de "tr",
      tio estigas efektivan skribon al la periferiaparato.
LFSKRIB ekvivalentas al LFSKRIB (ELIGO)
LFSKRIB (tr,pl,p2,...pn) ekvivalentas al :
      EK SKRIB(tr,pl); ......SKRIB(tr,pn); LFSKRIB (tr) FIN
LFSKRIB (pl, ...pn) ekvivalentas al :
      EK SKRIB(pl); ..... SKRIB (pn); LFSKRIB FIN
                aŭ ekvivalentas al :
      EK SKRIB(ELIGO,pl); ...SKRIB(ELIGO,pn); LFSKRIB(ELIGO) FIN
```

Jen la tri formoj kiujn povas havi parametron de subrutino SKRIB :

E

E : MKL

E : MKL : LFP

Klarigoj :

E : skribenda ekspresio kun tipo karaktra (KAR), reela (REEL), entjera (ENT), bulea (BUL) aŭ ĉena (CHEN).

MKL : (ne deviga). Minimuma KampLongo en kiu la rezulto de la ekspresio devas eniri. Se MKL estas tro malgranda por la rezulto, neniu stumpigo okazas, la tuta rezulto vidiĝas. Sen indiko de MKL, ekzistas implicaj kamplongoj (por ekzemplo : 20 por reelo, 12 por entjero, ekzakta longo por ĉeno). Se "E" estas bulea, la skribita valoro estas VERA aŭ FALSA.

LFP : (ne deviga). Longo de la Frakcia Parto (nur por la reeloj).

Por ekzemplo :

```
SKRIB (12,135.4) skribigas -->..........12........1.354000E+02
SKRIB ('KVADRAT = ',KV(5):4) skribigas --> KVADRAT = ..25
SKRIB(b=a,' ':5,'-',317.9:8:2) skribigas --> VERA.....-..317.90
```

Jen simpligoj pri la implicaj tekstaj rikordaroj :

SKRIB (k)	anstataŭ	SKRIB(ELIGO,k)
LEG (k)	"	LEG (ENIGO,k)
LFSKRIB	"	LFSKRIB (ELIGO)
LEGNL	"	LEGNL (ENIGO)
FDR	"	FDR (ENIGO)
FDL	"	FDL (ENIGO)

III.4.12. apartaj operacioj pri la ĉenoj.

Iuj kompileroj akceptas tiun tipon kun ĉiuj aŭ nur kelkaj el la sekvantaj funkcioj kaj subrutinoj.

Funkcio LONG : LONG(v) liveras la longecon de la ĉeno "v" (nombro de karaktroj en la variablo "v").

Funkcio POZ : POZ(v1,v2) liveras la pozicion de la unua enesto de la ĉeno v2 en la ĉeno v1.

Funkcio APUD : APUD(v1,v2,...vn) liveras ĉenon kiu konsistas el la orda apudmeto de la ĉenoj v1, v2 ĝis vn. Iuj kompileroj ne akceptas pli ol du parametrojn.

Funkcio KOPI : KOPI(v,p,l) liveras vicon kiu estas kopio de ĉenparto troviĝanta en la ĉeno "v" je la pozicio "p" kun longeco "l".

Subrutino ENCHEN : ENCHEN (fc, cc, p) estigas enĉenigon de la fontĉeno "fc" en la celĉenon "cc" je la pozicio "p".

Subrutino ELCHEN : ELCHEN (v,p,l) estigas elĉenigon el la ĉeno "v" de ĉenparto kun longeco "l" troviĝanta je la pozicio "p" en la ĉeno "v".

Por ekzemplo :

```
    p1 := 'TIO ESTAS';
    p2 := 'FRAZO';
    fr := APUD(p1,' '); % aldono de spaco %
    fr := APUD(fr,p2);
  Rezulto en fr : 'TIO ESTAS FRAZO'.
```

Ankaŭ la rilataj operatoroj = <= < > >= <> aplikiĝas al la ĉenaj variabloj. La ordo estas tiu de la ordo de la karaktro (vidi anekson 4).

Por ekzemplo :

'ETAK' < 'ETIK' liveras VERA

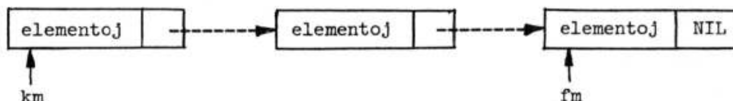
'ETAK ' < 'ETAKA' liveras FALSA

ĉar spaco situas post la literoj en la vico de la karaktroj.

III.4.13. apartaj operacioj pri la montriloj kaj la dinamikaj variabloj.

Ni disponas pri la subrutino NOV por krei dinamikan variablon.

Por ekzemplo : kreo de ligita listo :




```
TIP mtril = @elem;
    elem = RIK r: REEL;
               b: BUL;
               m: mtril
           FIN
VAR km, nm, fm : mtril; % deklaro de montriloj %
    % komenca, novelementa kaj fina montriloj %
```

Malplena ligita listo : km := NIL;

La elementoj de la listo ankoraŭ ne ekzistas, ni kreas ilin per la subrutino NOV.

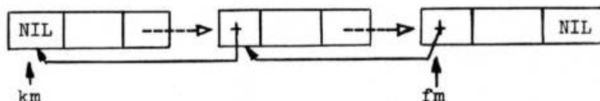
```
NOV(nm);
nm@.r := 2.5; nm@.b := VERA;
nm@.m := km;
km := nm; fm :=nm;
```

La ligita listo nun havas unu elementon, ni kreas novan kiun ni ligas al la ligita listo km :

```
NOV(nm);
nm@.r := 3.5; nm@.b := FALSA;
nm@.m := km;
km := nm;
```

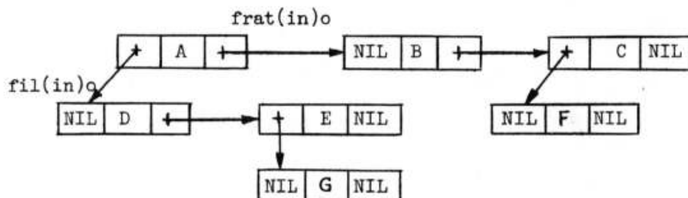
La ligita listo nun enhavas du elementojn.

Ni povas krei kompleksajn strukturojn, jen duligila listo :




```
TIP mtr = ∂ell;
    ell = RIK am: mtr; % antaŭa montrilo %
              a,b: ENT;
              pm: mtr % posta montrilo %
          FIN
VAR km, nm, fm : mtr;
```

Jen arbeca listo (aŭ familia listo) :



La deklaro de la tipoj por la duligila listo same validas por la arbeca listo.

Se montrilo "m" havas la valoron NIL, la variablo "m∂" ne ekzistas kaj uzo de "m∂" en tia kazo estigos fatalan eraron.

Ne forgesu initi la montrilajn variablojn je NIL, ne ĉiuj kompileroj faras tion por la programistoj.

La formo NOV(m,s1,s2,...sn) povas esti uzata por akiri memoron sufiĉan por rikordo kies elementoj havus la valoron "s1" ĝis "sn" sed tio ne atribuigas la valorojn "s1" ĝis "sn" al la diversaj kampoj de la nova variablo.

Se variablo post kreo kaj uzo ne plu estas necesa, ni povas redoni ĝin al la sistemo per la subrutino REDON : REDON(m) aŭ REDON(m, s1, s2,...sn) depende de la subrutino uzita por krei tiun variablon.

IV. EKZEMPLOJ DE PASCAL-AJ PROGRAMOJ.

Jen kelkaj ekzemploj de PASCAL-aj programoj por solvi matematikajn problemojn aŭ por ludi.

IV.1. la perfektaj nombroj.

Perfekta nombro karakterizîĝas per la fakto ke ĝi egalas al la sumo de siaj dividantoj escepte de si mem. Jen ekzemplo de perfekta nombro : 6 = 1 + 2 + 3.

Jen programo kiu trovas la perfektajn nombrojn kiuj ekzistas inter 1 kaj 1000 :

```
PROGRAM perfekta;
VAR n,i,j : ENT;
    sum : ENT;
VAL n = 1000;
EK
  POR i:=2 AL n FARI
  EK
    sum := 0;
    POR j:=1 AL i DIV 2 FARI
    EK
      SE i/j=i DIV j DO sum:= sum+j
    FIN;
    SE sum=i DO LFSKRIB(i,' estas perfekta nombro')
  FIN;
FIN.
```

IV.2. la negativaj potencoj de 2.

Ni deziras komputi la negativajn potencojn de 2 kaj havi ĉiujn utilajn decimalojn, tion ni ne povas fari uzante reelan variablon ĉar la maksimuma nombro de decimaloj estas 6 aŭ 7 sed ne pli. En tiu programo, ni konsideros la frakcian parton kiel ciferan tabelon kaj ni faros sinsekvajn dividojn kiel ni devas fari kiam ni kalkulas tion sen maŝino.

```
PROGRAM negpot;
TIP cifer = 0..9;
VAR n,i,j,r : ENT;
    d : TAB (.1..50.) DE cifer;
EK
  LEG(n);
  POR j:=1 AL n FARI
  EK
    SKRIB('negativa potenco de 2 je ',j:2,' : .');
    POR i:=1 AL k-1 FARI
    EK
      r:= 10 + r + d(.i.);
      d(.i.):= r DIV 2; r:= r - 2 + d(.i.);
      SKRIB( KR( d(.i.) + RANG( '0' ) ) )
    FIN;
    d(.j.):= 5;
    LFSKRIB('5')
  FIN;
FIN.
```

Jen la skribita rezulto de la efektivigo de tiu programo :

```
negativa potenco de 2 je  1 : .5
negativa potenco de 2 je  2 : .25
negativa potenco de 2 je  3 : .125
.............
```

IV.3. la nombroj de Fibonacci.

La nombroj de Fibonacci (elp. fibonaĉi) estas la solvoj de la sekvanta formulo : $F_{n+2} = F_{n+1} + F_n$ kie $F_0 = 0$ kaj $F_1 = 1$ kun "n" pozitiva aŭ nula.

```
PROGRAM fibonacci;
VAR i,n,fn0,fnl,fn2 : ENT;
EK
  fn0:=0; fnl:=l;
  LEG(n);
  LFSKRIB('jen la ',n+2:2,' unuaj nombroj de Fibonacci');
  LFSKRIB('0');
  LFSKRIB('1');
  POR i:=l AL n FARI
  EK
    fn2:= fnl + fn0; fn0:= fnl; fnl:= fn2;
    LFSKRIB(fn2);
  FIN;
FIN.
```

Jen la 7 unuaj nombroj de Fibonacci : 0, 1, 1, 2, 3, 5, 8...

Jen alia maniero por solvi tiun problemon (per uzo de funkcio).

```
PROGRAM snf; % Skribo de la Nombroj de Fibonacci %
FUNKCI fibonacci(n:ENT):ENT;
VAR i,fn0,fnl: ENT;
EK
  fn0:=0; fnl:=l;
  i:=l;
  DUM i < n FARI
  EK
    fnl := fnl + fn0;
    fn0 := fnl - fn0;
    i := i + l
  FIN;
  fibonacci := fnl
FIN;
```

```
EK  % komenco de la programo %
  LFSKRIB('jen la ',n+l,' unuaj nombroj de Fibonacci');
  LFSKRIB('0');
  LEG(n);
  POR i:=1 AL n FARI
  LFSKRIB(fibonacci(i))
FIN.
```

IV.4. komptado de la frekvenco de literoj en teksto.

Ni analizos tekston kaj ni komptos por ĉiuj literoj kiom da fojoj ili aperas en la teksto.

```
PROGRAM litfrek; %litera frekvenco %
VAR k: KAR;
    kompt: TAB (.'a'..'z'.) DE ENT;
    liter: AR DE 'a'..'z';
EK
  liter :  (.'a'..'z'.);
  POR k:='a' AL 'z' FARI kompt(.k.):=0;
  DUM NE FDR FARI
  EK
    DUM NE FDL FARI
    EK
      LEG(k); SKRIB(k);
      SE k EN liter DO kompt(.k.) := kompt(.k.) + 1
    FIN;
    LFSKRIB; LEGNL;
  FIN;
FIN.
```

IV.5. la ok damoj.

Sur ŝaktabulo, eblas lokigi 8 damojn sen ke iu minacu alian. En la ŝakludo, la damo minacas ĉiujn fakojn sur la samaj linio kaj kolumno (kiel turo) kaj ĉiujn fakojn laŭ la du diagonaloj (kiel kuriero). Tiu problemo havas 92 solvojn.

```
PROGRAM damoj;
KONST n=8;
VAR kol: TAB (.1..n.) DE ENT;
    nsol,lini,kolumno,l: ENT;
FUNKCI konflikt(k,l:ENT): BUL;
VAR dl,kl: ENT;
EK
  konflikt:= FALSA;
  POR kl:=1 AL k FARI
  EK
    dl:= ABS(kol(.kl.)-l);
    SE dl+(dl-k-l+kl) = 0 DO    konflikt:= VERA;
  FIN;
FIN;
SUBRUTIN solvo(VAR sol: ENT);
VAR j,k : ENT;
EK
  sol:=sol+1;
  LFSKRIB('solvo no ',sol);
  POR j:=1 AL n FARI
  EK
    POR k:=1 AL n FARI
      SE k=kol(.j.) DO SKRIB('D') SENE SKRIB('+');
    LFSKRIB;
  FIN;
FIN;
% ĉefa programo %
EK
  nsol:=0;
  POR l:=1 AL n FARI
  EK
    kol(.l.):=l; kolumno:=1; lini:=1;
    DUM kolumno  <>  0 FARI
    EK
      DUM lini  <= 8 FARI
        SE konflikt(kolumno,lini) DO lini:=lini+1
        SENE
        EK
          kol(.kolumno 1.):=lini;
          kolumno:=kolumno+1;
```

```
            SE kolumno <= n-1 DO lini:=1
            SENE
            EK
              solvo(nsol); lini:=n+1
            FIN;
          FIN;
        DUM (lini > 8) KAJ (kolumno <> 0) FARI
        EK
          kolumno:= kolumno-1;
          lini := kol(.kolumno 1.)+1;
        FIN;
      FIN;
    FIN;
  FIN.
```

Jen la tri unuaj solvoj de tiu problemo :

solvo no 1	solvo no 2	solvo no 3
D + + + + + + +	D + + + + + + +	+ D + + + + + +
+ + + + D + + +	+ + + + + D + +	+ + + + + + D +
+ + + + + + + D	+ + + + + + + D	+ + + D + + + +
+ + + + + D + +	+ + D + + + + +	+ + + + + D + +
+ + D + + + + +	+ + + + + + D +	+ + + + + + + D
+ + + + + + D +	+ + + D + + + +	+ D + + + + + +
+ D + + + + + +	+ D + + + + + +	+ + + + D + + +
+ + + D + + + +	+ + + + D + + +	+ + D + + + + +

Anekso 1 : Rezervitaj vortoj kaj normigitaj identigiloj.

1. rezervitaj vortoj.

1.1. rezervitaj vortoj laŭ la angla alfabetordo.

angle/france/esperante	angle/france/esperante
AND/ET/KAJ	NOT/NON/NE
ARRAY/TABLEAU/TAB(ELO)	OF/PARMI, DE/INTER, DE
BEGIN/DEBUT/EK	OR/OU/AU
CASE/CAS/KAZ(O)	*ORIGIN/ORIGINE/ORIGIN(O)
CONST/CONST/KONST(ANTO)	OTHERWISE/AUTREMENT/ALIE
DIV/DIV/DIV(IDI)	PACKED/COMPACT/KOMP(AKT)A
DO/FAIRE/FARI	PROCEDURE/PROCEDURE/SUBRUTIN(O)
DOWNTO/DECR ou D/SUBAL	PROGRAM/PROGRAMME/PROGRAM(O)
ELSE/SINON/SENE	RECORD/STRUCT/RIK(ORDO)
END/FIN/FIN(O)	REPEAT/REPETER/RIPETI
*EXIT/SORTIE/ELIR(O)	SET/ENSEMBLE/AR(O)
FILE/FICHIER/RIK(ORD)AR(O)	*STRING/CHAINE/CHEN(O)
FOR/POUR/POR	THEN/ALORS/DO
FORWARD/AVANT/POSTE	TO/INCR ou A/AL
FUNCTION/FONCTION/FUNKCI(O)	TYPE/TYPE/TIP(O)
GOTO/ALLERA/IRIAL	UNTIL/JUSQUA/GHIS
IF/SI/SE	*VAL/VAL/VAL(ORO)
IN/DANS/EN	VAR/VAR/VAR(IABLO)
LABEL/ETIQUETTE/ETIK(EDO)	WHILE/TANTQUE/DUM
MOD/MOD/MOD(ULUSO)	WITH/AVEC/KUN
NIL/NIL/NIL	

La vortoj signitaj per asterikso ne estas rezervitaj por ĉiuj kompileroj.

La literoj inter krampoj ne apartenas al la rezervitaj vortoj, ili aperas en tiu listo nur por klarigi la rezervitajn vortojn.

1.2. rezervitaj vortoj laŭ la esperanta alfabetordo.

Esperante/angle/france

AL/TO/INCR ou A
ALIE/OTHERWISE/AUTREMENT
AR(O)/SET/ENSEMBLE
AU/OR/OU
*CHEN/STRING/CHAINE
DE/OF/DE
DO/THEN/ALORS
DIV(IDI)/DIV/DIV
DUM/WHILE/TANTQUE
EK/BEGIN/DEBUT
*ELIR(O)/EXIT/SORTIE
EN/IN/DANS
ETIK(EDO)/LABEL/ETIQUETTE
FARI/DO/FAIRE
FIN(O)/END/FIN
FUNKCI(O)/FUNCTION/FONCTION
GHIS/UNTIL/JUSQUA
INTER/OF/PARMI
IRIAL/GOTO/ALLERA
KAJ/AND/ET
KAZ(O)/CASE/CAS

esperante/angle/france

KOMP(AKT)A/PACKED/COMPACT
KONST(ANTO)/CONST/CONST
KUN/WITH/AVEC
MOD(ULUSO)/MOD/MOD
NE/NOT/NON
NIL/NIL/NIL
*ORIGIN(O)/ORIGIN/ORIGINE
POR/FOR/POUR
POSTE/FORWARD/AVANT
PROGRAM(O)/PROGRAM/PROGRAMME
RIK(ORDO)/RECORD/STRUCT
RIK(ORD)AR(O)/FILE/FICHIER
RIPETI/REPEAT/REPETER
SE/IF/SI
SENE/ELSE/SINON
SUBAL/DOWNTO/DECR ou D
SUBRUTIN(O)/PROCEDURE/PROCEDURE
TAB(ELO)/ARRAY/TABLEAU
TIP(O)/TYPE/TYPE
*VAL(ORO)/VAL/VAL
VAR(IABLO)/VAR/VAR

La vortoj signitaj per asterikso ne estas rezervitaj vortoj por ĉiuj kompileroj.

La literoj inter krampoj ne apartenas al la rezervitaj vortoj, ili aperas en tiu listo nur por klarigi la rezervitajn vortojn.

1.3. listo de la instrukcifrazoj laŭ la angla alfabetordo.

ARRAY ......OF	TAB ..... DE .....
BEGIN .....END	EK ......FIN
CASE ..... OF	KAZ .....INTER .....
CASE ... OF ... OTHERWISE ...END	KAZ ... INTER ... ALIE ... FIN
FILE OF	RIKAR DE .....
FOR ... DOWNTO ... DO ...	POR ... SUBAL ... FARI ...
FOR ... TO ... DO ...	POR ... AL ... FARI ...
IF ... THEN ... ELSE	SE ... DO ... SENE ...
PACKED ARRAY ... OF ...	KOMPA TAB ... DE ....
RECORD ... END	RIK ... FIN
REPEAT ... UNTIL ...	RIPETI ... GHIS ...
SET OF	AR DE ....
WHILE ... DO	DUM ... FARI ....
WITH ... DO ...	KUN ... FARI ...

1.4. listo de la instrukcifrazoj laŭ la esperanta alfabetordo.

AR DE ..... : SET OF ....
DUM ... FARI ... : WHILE ... DO ...
EK ... FIN : BEGIN ... END
KAZ ... INTER ... : CASE ... OF ...
KAZ ... INTER ... ALIE ... FIN : CASE ... OF ... OTHERWISE ... END
KOMPA TAB ... DE ... : PACKED ARRAY ... OF ...
KUN ... FARI ... : WITH ... DO ...
POR ... AL ... FARI ... : FOR ... TO ... DO ...
POR ... SUBAL ... FARI ... : FOR ... DOWNTO ... DO ...
RIK ... FIN : RECORD ... END
RIKAR DE ... : FILE OF ...
RIPETI ... GHIS ... : REPEAT ... UNTIL ...
SE ... DO ... SENE ... : IF ... THEN ... ELSE ...
TAB ... DE ... : ARRAY ... OF ...

2. La normigitaj identigiloj.

2.1. la konstantoj.

angle/france/esperante

ALFALENG/ALFALONG/ALFALONG : nombro de karaktroj en vortoj.
MAXINT/ENTMAX/MAKSENT : plej altranga entjero disponebla en la maŝino.
FALSE/FAUX/FALSA : bulea valoro.
TRUE/VRAI/VERA : la alia bulea valoro.

2.2. la rikordaroj por datenaj enigoj-eligoj.

angle/france/esperante

INPUT/ENTREE/ENIGO : plej kutima eniga rikordaro (kartlegilo).
OUTPUT/SORTIE/ELIGO : plej kutima eliga rikordaro (printero).

2.3. la tipoj.

angle/france/esperante

INTEGER/ENTIER/ENT(JERO) : entjero.
REAL/REEL/REEL(O) : reelo.
BOOLEAN/BOOLEEN/BUL(EAJHO) : tipo kun du eblaj valoroj : (FALSA-VERA).
CHAR/CAR/KAR : karaktro (ĉiuj disponeblaj karaktroj).
TEXT/TEXTE/TEKST(O) : teksta rikordaro.
ALFA/ALFA/ALFA : ĉeno de karaktroj kies longo estas ALFALONG.
STRING/CHAINE/CHEN : ĉeno de karaktroj.

ANEKSO 2 : LISTO DE LA JAM DIFINITAJ SUBRUTINOJ.

angle/france/esperante	priskribo de la subrutinoj.
DELETE/EFFACER/ELCHEN(c,p,l):	Elĉenigo de ĉenparto kun longeco "l" je pozicio "p" en la ĉeno "c".
DISPOSE/LIBERER/REDON(m):	Dinamika liberigo de montrilo.
GET/PRENDRE/AKIR(r):	Akiro de rikordo el la rikordaro "r" en la bufra variablo "ro". Se ĝi ne ekzistas FDR(r) fariĝas VERA kaj "ro" havas nekonatan valoron. Tiu subrutino estas vokebla nur se FDR(r) estas FALSA.
INSERT/INSERER/ENCHEN(fc,cc,p):	Enĉenigo de la ĉeno "fc" en la ĉenon "cc" je la pozicio "p".
NEW/NOUVEAU/NOV(m)	Dinamika kreo de nova variablo de la tipo ligita al "m" kaj atribuas la adreson de tiu variablo al "m".
PACK/COMPACTER/KOMP(t,i,a):	Kompaktigo de la variablo "t" en "a".
PAGE/PAGE/PAGH(r):	Paĝsalto en la rikordaro "r".
PUT/METTRE/MET(r):	Skribo de la valoro de la bufra variablo "ro" en la rikordaron "r". FDR(r) devas esti VERA antaŭe kaj post tio restas VERA.
READ/LIRE/LEG(v):	Lego de la variablo "v" sen iro komence de nova linio.
READLN/LIRELN/LEGNL(v):	Lego de la variablo "v" kaj poste iro komence de nova linio.
RESET/RELIRE/LRES(r)	Preparo de la rikordaro "r" por dekomenca lego (LegREStarto). Se "r" ne estas malplena, FDR(r) fariĝas FALSA. Deviga ordono antaŭ la unua lego (escepte por "ENIGO").
REWRITE/REECRIRE/SRES(r):	Preparo de la rikordaro "r" por dekomenca skribo (SkribREStarto). "r" fariĝas malplena kaj FDR(r) fariĝas VERA. Deviga ordono antaŭ la unua skribo (escepte por "ELIGO").
UNPACK/DILATER/MKOMP(t,a,i):	Malkompaktigo de la variablo "t" en "a".
WRITE/ECRIRE/SKRIB(v):	Skribo de la variablo "v".
WRITELN/ECRIRELN/LFSKRIB(v):	Skribo de la variablo "v" kaj poste de linifino.

ANEKSO 3 : LISTO DE LA JAM DIFINITAJ FUNKCIOJ.

Angle/france/esperante	Priskribo de la funkcioj.
ABS/ABS/ABS(v)	Absoluta valoro de "v" : 'v'
ARCTAN aŭ ATAN//ATAN(v)	Arktangento de "v"
CHR/CARAC/KR(v)	"v"-a karaktro. "v" estas entjertipa, la rezulto estas karaktro kies rango indikas "v" se ĝi ekzistas.
CONCAT//APUD(v1,...vn)	Apudmeto de v1,...vn ĉenoj en unu ĉenon.
COPY/COPIE/KOPI(c,p,l)	Liveras ĉenparton de ĉeno "c" komenciĝanta je pozicio "p" kun longeco "l".
COS/COS/KOS(v)	Kosinuso de angulo (en arkuso).
EOF/FDF/FDR(r)	Fino de rikordaro (rezulto: VERA(fino) aŭ FALSA(ne fino)).
EOLN/FDL/FDL(r)	Fino de linio en teksta rikordaro : (rezulto : VERA(fino) aŭ FALSA(ne fino de linio)).
EXP/EXP/EKSP(v)	e : eksponencialo de "v".
LENGTH/LONG/LONG(v)	Longo de la ĉeno "v".
LN/LN/NL(v)	Nepera logaritmo de "v".
ODD/IMPAIR/MPAR(v)	Malpareco de "v" (VERA-FALSA).
ORD/RANG/RANG(v)	Rango de "v". "v" estas skalartipa, la rezulto estas la rango (entjero) de la valoro de "v" en la aro difinita per la tipo de "v".
POS/POS/POZ(c1,c2)	Pozicio de la ĉeno c2 en la ĉeno c1.
PRED/PRED/ANT(v)	Valoro antaŭ la valoro de "v" se ĝi ekzistas en la tipo de "v". "v" estas skalartipa sed ne reeltipa.
ROUND/ARRONDI/ROND(v)	Rondigo de "v". La rezulto estas la entjero la plej proksima al la valoro de "v".
SIN/SIN/SIN(v)	Sinuso de angulo "v" (en arkuso).
SQR/CARRE/KV(v)	Kvadrato de "v" aŭ "v" je potenco 2.
SQRT/RAC2/KVR(v)	Kvadrata radiko de "v".
SUCC/SUCC/POST(v)	Valoro post "v" se ĝi ekzistas en la tipo de "v". "v" estas skalartipa sed ne reeltipa.
TRUNC/TRONC/TRUNK(v)	Trunko de "v". La rezulto estas la entjera parto de reelo "v".

ANEKSO 4 : TABELO DE LA KARAKTROJ.

				b7	0	0	0	0
				b6	0	0	1	1
				b5	0	1	0	1
b4	b3	b2	b1		0	1	2	3
0	0	0	0	0	"	Pp	5	'
0	0	0	1	1	Aa	Qq	6	[
0	0	1	0	2	Bb	Rr	7	]
0	0	1	1	3	Cc	Ss	8	:
0	1	0	0	4	Dd	Tt	9	#
0	1	0	1	5	Ee	Uu	+	%
0	1	1	0	6	Ff	Vv	-	!
0	1	1	1	7	Gg	Ww	*	&
1	0	0	0	8	Hh	Xx	/	@
1	0	0	1	9	Ii	Yy	(	_
1	0	1	0	A	Jj	Zz	)	<
1	0	1	1	B	Kk	0	$	>
1	1	0	0	C	Ll	1	=	?
1	1	0	1	D	Mm	2		^
1	1	1	0	E	Nn	3	,	|
1	1	1	1	F	Oo	4	.	;

	0	1	2	3	4	5	6	7	8	9
0	"	Aa	Bb	Cc	Dd	Ee	Ff	Gg	Hh	Ii
1	Jj	Kk	Ll	Mm	Nn	Oo	Pp	Qq	Rr	Ss
2	Tt	Uu	Vv	Ww	Xx	Yy	Zz	0	1	2
3	3	4	5	6	7	8	9	+	-	*
4	/	(	)	$	=		,	.	'	[
5	]	:	#	%	!	&	@	_	<	>
6	?	^	|	;						

ANEKSO 5 : LA SINTAKSO DE LA PROGRAMLINGVO "PASCAL".

Arbitre, ni decidis skribi la rezervitajn vortojn kaj identigilojn per majuskloj por pli bone apartigi ilin de la teksto mem.

Ni adoptis la sekvantajn konvenciojn (BNF: Backus-a Norma Formo) por priskribi la sintakson de la programlingvo :

"_____" fina elemento kiu tia aperas en la teksto de la programo mem.
<_____> limigas nefinan elenenton nomatan "sintaksa variablo".
__::=__ signifas "estas difinita kiel".
{_____} limigas opcian aŭ ripetiĝan elementon, tio estas elemento kiu povas ne enesti aŭ unuope aŭ plurope enesti.
___|___ signifas "aŭ".
[_____] limigas opcian elementon, tio estas elemento kiu povas ne enesti aŭ nur unuope enesti.
__()___ malplena aro (nenio).

La radiko de la gramatiko estas <programo>.

<adicia operatoro> ::= "+" | "-" | "AU"
<alvariabla referenco> ::= <kompleta variablo> | <era variablo> | <montrata variablo> | <bufra variablo>
<ara konsistigilo> ::= "[" <reprezentilo de elemento> {,<reprezentilo de elemento>} "]"
<ara tipo> ::= "AR" "DE" <baza tipo>
<atribua instrukcifrazo> ::= <alvariabla referenco> ":=" <ekspresio> | < funkciidentigilo> ":=" <ekspresio>
<baza tipo> ::= <skalara tipo>
<bloko> ::= <etikedodeklara parto> <konstantdifina parto> <tipdifina parto> <variablodeklara parto> <subrutindeklara parto> <funkcideklara parto> <instrukciteksta parto>
<bufra variablo> ::= <rikordara variablo> "^"
<bulea ekspresio> ::= <ekspresio>
<cifera vico> ::= <cifero> { <cifero> }
<cifero> ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
<ĉena karaktro> ::= iu ajn videbla ASCII-a karaktro
<ĉena tipo> ::= "CHEN" "["<maksimuma longo>"]"
<ĉenero> ::= "''" | <ĉena karaktro>
<karaktra ĉeno> ::= "'" <ĉenero> { <ĉenero> } "'"
<deksesuma cifera vico> ::= <deksesuma cifero> {<deksesuma cifero> }

```
<deksesuma cifero> ::= <cifero> | "A" | "B" | "C" | "D"
        | "E" | "F"
<direktivo> ::= <litero> { <litero> | <cifero> }
<duuma cifera vico> ::= <duuma cifero> { <duuma cifero>}
<duuma cifero> ::= "0" | "1"
<efektiva parametro> ::= <ekspresio> | <alvariabla
        referenco> | <subrutinidentigilo> |
        <funkciidentigilo>
<ekspresio> ::= <simpla ekspresio> [ <rilatiga
        operatoro> <simpla ekspresio> ]
<elemento de kaza listo> ::= <listo de kazaj konstantoj>
        ":" <instrukcibloko>
<era tipo> ::= <tipdeskriptoro>
<era variablo> ::= <indicita variablo> | <kampa
        reprezentilo>
<etikedo> ::= <cifera vico>
<etikedodeklara parto> ::= [ "ETIK" <etikedo> { ","
        <etikedo> } ";" ]
<faktoro> ::= <alvariabla referenco> | <sensigna
        konstanto> | <limiga identigilo> | <funkcia
        reprezentilo> | <ara konsistigilo> | "("
        <ekspresio> ")" | "NE" <faktoro>
<fiksita parto> ::= <rikorda sekcio> { ";" <rikorda
        sekcio> }
<fina valoro> ::= <ekspresio>
<formatfaktoro> ::= <kunsigna entjero>
<frakcia parto> ::= <cifera vico>
<funkcia identigo> ::= "FUNKCI" <funkciidentigilo>
<funkcia reprezentilo> ::= <funkciidentigilo> [ <listo
        de efektivaj parametroj> ]
<funkcibloko> ::= <bloko>
<funkcideklara parto> ::= <funkcikapumo> ";" <direktivo>
        | <funkcia identigo> ";" <funkcibloko> |
        <funkcikapumo> ";" <funkcibloko>
<funkciidentigilo> ::= <identigilo>
<funkcikapumo> ::= "FUNKCI" <identigilo> [ <listo de
        formalaj parametroj> ] ":" <tipo de rezulto>
<identigila listo> ::= <identigilo> { "," <identigilo> }
<identigilo> ::= <litero> { <litero> | <cifero> }
<identigilo de kampa reprezentilo> ::= <identigilo>
<identigilo de montrila tipo> ::= <tipidentigilo>
<identigilo de reela tipo> ::= <tipidentigilo> | "REEL"
<identigilo de simpla tipo> ::= <tipidentigilo>
<identigilo de skalara tipo> ::= <tipidentigilo> | "BUL"
        | "REEL" | "KAR" | "ENT"
<identigilo de struktura tipo> ::= <tipidentigilo>
<indica ekspresio> ::= <ekspresio>
<indica tipo> ::= <skalara tipo>
<indicita variablo> ::= <tabela variablo> "[" <indica
        ekspresio> { "," <indica ekspresio> } "]"
<indikila kampo> ::= <identigilo>
<indikila tipo> ::= <identigilo de skalara tipo>
<instrukcibloka vico> ::= <instrukcibloko> { ";"
        <instrukcibloko> }
<instrukcibloko> ::= [ <etikedo> ":" ] <simpla
        instrukcifrazo> | [ <etikedo> ":" ] <struktura
        instrukcibloko>
<instrukcifrazo DUM> ::= "DUM" <bulea ekspresio> "FARI"
        <instrukcibloko>
```

```
<instrukcifrazo IRIAL> ::= "IRIAL" <etikedo>
<instrukcifrazo KAZ> ::= "KAZ" <kaza indico> "INTER"
          <elemento de kaza listo> { ";" <elemento de
          kaza listo> } [ ";" ] "FIN"
<instrukcifrazo KUN> ::= "KUN" <listo de rikordaj
          variabloj> "FARI" <instrukcibloko>
<instrukcifrazo POR> ::= "POR" <regvariablo> ":="
          <komenca valoro> "AL" <fina valoro> "FARI"
          <instrukcibloko> | "POR" <regvariablo> ":="
          <komenca valoro> "SUBAL" <fira valoro> "FARI"
          <instrukcibloko>
<instrukcifrazo RIPETI> ::= "RIPETI" <instrukcibloka
          vico> "GHIS" <bulea ekspresio>
<instrukcifrazo SE> ::= "SE" <bulea ekspresio> "DO"
          <instrukcibloko> [ <parto SENE> ]
<instrukciteksta parto> ::= <plurera instrukcibloko>
<intervala tipo> ::= <konstanto> ".." <konstanto>
<kampa listo> ::= <fiksita parto> ! <fiksita parto> ";"
          <varia parto> ! <varia parto>
<kampa reprezentilo> ::= <rikorda variablo> "."
          <kampidentigilo> <identigilo de kampa
          reprezentilo>
<kampidentigilo> ::= <identigilo>
<karaktra ĉeno> ::= "'" <ĉenero> { <ĉenero> } "'"
<kaza indico> ::= <ekspresio>
<kaza konstanto> ::= <konstanto>
<komenca valoro> ::= <ekspresio>
<kompleta variablo> ::= <variabla identigilo>
<kondiĉa instrukcifrazo> ::= <instrukcifrazo SE> !
          <instrukcifrazo KAZ>
<konstantdifina parto> ::= [ "KONST" <<onstantdifino>
          ";" { <konstantdifino> ";" } ]
<konstantdifino> ::= <identigilo> "=" <konstanto>
<konstantidentigilo> ::= <identigilo>
<konstanto> ::= [ <signo> ] <sensigna nombro> ! [
          <signo> ] <konstantidentigilo> ! <karaktra
          ĉeno>
<kunripeta instrukcifrazo> ::= <instrukcifrazo RIPETI> !
          <instrukcifrazo DUM> ! <instrukcifrazo POR>
<kunsigna entjero> ::= [ <signo> ] <sensigna entjero>
<kunsigna reelo> ::= [ <signo> ] <sensigna reelo>
<limiga identigilo> ::= <identigilo>
<lista tipo> ::= "(" <identigila listo> ")"
<listo de efektivaj parametroj> ::= "(" <efektiva
          parametro> { "," <efektiva parametro> } ")"
<listo de formalaj parametroj> ::= "(" <sekcio de
          formalaj parametroj> { ";" <sekcio de formalaj
          parametroj> } ")"
<listo de kazaj konstantoj> ::= <kaza konstanto> { ","
          <kaza konstanto> }
<listo de rikordaj variabloj> ::= <rikorda variablo> {
          "," <rikorda variablo> }
<litero> ::= "A"!"B"!"C"!"D"!"E"!"F"!"G"!"H"!"I"!"J"!
          "K"!"L"!"M"!"N"!"O"!"P"!"Q"!"R"!"S"!"T"!"U"!"V"!

          "W"!"X"!"Y"!"Z"!"a"!"b"!"c"!"d"!"e"!"f"!"g"!"h"!

          "i"!"j"!"k"!"l"!"m"!"n"!"o"!"p"!"q"!"r"!"s"!"t"
          ! "u"!"v"!"w"!"x"!"y"!"z"!"_"
```

<maksimuma longo> ::= <sensigna entjero>

<malplena instrukcifrazo> ::= ()

<modelo de alĝustigebla tabelo> ::= <modelo de nekompakta alĝustigebla tabelo> ! <modelo de kompakta alĝustigebla tabelo>

<modelo de kompakta alĝustigebla tabelo> ::= "KOMPA" "TAB" "[" <priskribo de indica tipo> "]" "DE" <tipidentigilo>

<modelo de nekompakta alĝustigebla tabelo> ::= "TAB" "[" <priskribo de indica tipo> { ";" <priskribo de indica tipo> } "]" "DE" <tabelera priskribo>

<montrata variablo> ::= <montrila variablo> "↑"

<montrila tipo> ::= <nova montrila tipo> ! <identigilo de montrila tipo>

<montrila variablo> ::= <alvariabla referenco>

<multiplika operatoro> ::= "*" ! "/" ! "DIV" ! "MOD" ! "KAJ"

<nekompakta struktura tipo> ::= <tabela tipo> ! <rikorda tipo> ! <ara tipo> ! <rikordara tipo> ! <ĉena tipo>

<nova montrila tipo> ::= "↑" <tipidentigilo>

<nova tipo> ::= <nova skalara tipo> ! <nova struktura tipo> ! <nova montrila tipo>

<nova skalara tipo> ::= <lista tipo> ! <intervala tipo>

<nova struktura tipo> ::= ["KOMPA"] <nekompakta struktura tipo>

<okuma cifera vico> ::= <okuma cifero> { <okuma cifero>}

<okuma cifero> ::= "0" ! "1" ! "2" ! "3" ! "4" ! "5" ! "6" ! "7"

<parto SENE> ::= "SENE" <instrukcibloko>

<plurera instrukcibloko> ::= "EK" <instrukcibloka vico> "FIN"

<priskribo de alĝustigeblaj tabeloj: valoroj> ::= <identigila listo> ":" <modelo de alĝustigebla tabelo>

<priskribo de alĝustigeblaj tabeloj: variabloj> ::= "VAR" <identigila listo> ":" <modelo de alĝustigebla tabelo>

<priskribo de indica tipo> ::= <identigilo> ".." <identigilo> ":" <identigilo de skalara tipo>

<priskribo de parametro: funkcio> ::= <funkcikapumo>

<priskribo de parametro: subrutino> ::= <subrutinkapumo>

<priskribo de parametroj: alĝustigeblaj tabeloj> ::= <priskribo de alĝustigeblaj tabeloj: valoroj> ! <priskribo de alĝustigeblaj tabeloj: variabloj>

<priskribo de parametroj: valoroj> ::= <identigila listo> ":" <tipidentigilo>

<priskribo de parametroj: variabloj> ::= "VAR" <identigila listo> ":" <tipidentigilo>

<programbloko> ::= <bloko>

<programkapumo> ::= "PROGRAM" <identigilo> ["(" <programparametroj> ")"]

<programo> ::= <programkapumo> ";" <programbloko> "."

<programparametroj> ::= <identigila listo>

<regvariablo> ::= <kompleta variablo>

<reprezentilo de elemento> ::= <ekspresio> [".." <ekspresio>]

<rezervita vorto> ::= "KAJ" ! "TAB" ! "EK" ! "KAZ" ! "KONST" ! "CHEN" ! "DIV" ! "FARI" ! "SUBAL" ! "ALIE" ! "FIN" ! "RIKAR" ! "FUNKCI" ! "IRIAL" ! "SE" ! "EN" ! "ETIK" ! "MOD" ! "NIL" ! "NE" !

```
          "DE" ! "INTER" ! "AU" ! "KOMPA" ! "SUBRUTIN" !
          "PROGRAM" ! "RIK" ! "RIPETI' ! "AR" ! "DO" !
          "AL" ! "TIP" ! "GHIS" ! "VAR" ! "VAL" ! "DUM" !
          "KUN"
<rikorda sekcio> ::= <identigila listo> ":"
          <tipdeskriptoro>
<rikorda tipo> ::= "RIK" <kampa listo> "FIN"
<rikorda variablo> ::= <alvariabla referenco>
<rikordara tipo> ::= "RIKAR" "DE" <era tipo>
<rikordara variablo> ::= <alvariabla referenco>
<rilatiga operatoro> ::= "=" ! "<>" ! "<" ! ">" ! "<=" !
          ">=" ! "EN"
<sekcio de formalaj parametroj> ::= <priskribo de
          parametroj : valoroj> ! <priskribo de
          parametroj: variabloj> ! <priskribo de
          parametro: subrutino> ! <priskribo de
          parametro: funkcio> ! <priskribo de parametroj:
          alĝustigeblaj tabeloj>
<sensigna entjero> ::= "2# <duuma cifera vico> ! "8#
          <okuma cifera vico> ! [ "10#] <cifera vico> !
          "16# <deksesuma cifera vico>
<sensigna konstanto> ::= <sensigna nombro> ! <karaktra
          ĉeno> ! <konstanta identigilo> ! "NIL"
<sensigna nombro> ::= <sensigna entjero> ! <sensigna
          reelo>
<sensigna reelo> ::= <sensigna entjero> "." <frakcia
          parto> [ "E" <formatfaktoro> ] ! <sensigna
          entjero> "E" <formatfaktoro>
<signo> ::= "+" ! "-"
<simpla ekspresio> ::= [ <signo> ] <termo> { <adicia
          operatoro> <termo> }
<simpla instrukcifrazo> ::= <malplena instrukcifrazo> !
          <atribua instrukcifrazo> ! <subrutinvoka
          instrukcifrazo> ! <instrukcifrazo IRIAL>
<simpla tipo> ::= <skalara tipo> ! <tipidentigilo>
<skalara tipo> ::= <nova skalara tipo> ! <identigilo de
          skalara tipo>
<speciala simbolo> ::= "+" ! "-" ! "*" ! "/" ! "=" ! "<"
          ! ">" ! "[" ! "]" ! "." ! ," ! ";" ! ":" ! "↑"
          ! "(" ! ")" ! "<>" ! "<=" ! ">=" ! ":=" ! ".."
          ! <rezervita vorto>
<struktura instrukcibloko> ::= <plurera instrukcibloko>
          ! <kondiĉa instrukcifrazo> ! <kunripeta
          instrukcifrazo> ! <instrukcifrazo KUN>
<struktura tipo> ::= <nova struktura tipo> ! <identigilo
          de struktura tipo>
<subrutinidentigo> ::= "SUBRUTIN" <subrutinidentigilo>
<subrutinbloko> ::= <bloko>
<subrutindeklara parto> ::= <subrutinkapumo> ";"
          <direktivo> ! <subrutinidentigo> ";"
          <subrutinbloko> ! <subrutinkapumo> ";"
          <subrutinbloko>
<subrutinidentigilo> ::= <identigilo>
<subrutinkapumo> ::= "SUBRUTIN" <identigilo> [ <listo de
          formalaj parametroj> ]
<subrutinvoka instrukcifrazo> ::= <subrutinidentigilo> [
          <listo de efektivaj parametroj> ]
<tabela tipo> ::= "TAB" "[" <indica tipo> { "," <indica
          tipo> } "]" "DE" <era tipo>
```

<tabela variablo> ::= <alvariabla referenco>
<tabelera priskribo> ::= <tipidentigilo> ! <modelo de
    alĝustigebla tabelo>
<termo> ::= <faktoro> { <multiplika operatoro> <faktoro>}
<tipdeskriptoro> ::= <simpla tipo> ! <struktura tipo> !
    <montrila tipo>
<tipdifina parto> ::= ["TIP" <tipdifino> ";" {
    <tipdifino> ";" }]
<tipdifino> ::= <identigilo> "=" <tipdeskriptoro>
<tipidentigilo> ::= <identigilo>
<tipo de rezulto> ::= <identigilo de simpla tipo> !
    <identigilo de montrila tipo>
<varia parto> ::= "KAZ" <variantselektilo> "INTER"
    <varianto> { ";" <varianto> }
<variabla identigilo> ::= <identigilo>
<variablodeklara parto> ::= ["VAR" <variablodeklaro>
    ";" { <variablodeklaro> ";" }]
<variablodeklaro> ::= <identigila listo> ":"
    <tipdeskriptoro>
<varianto> ::= <listo de kazaj konstantoj> ":" "("
    <kampa listo> ")"
<variantselektilo> ::= [<indikila kampo> ":"]
    <indikila kampo>

Anekso 6 : naskiĝdatoj de la programadlingvoj.

jaro	ĝenerala	administra	scienca	aparta
1956			FORTRAN	IPL(komputora ŝarĝo)
1957		COMIT		
1958			ALGOL 58	IPL V
			FORTRAN II	
1959			MAD	DYNAMO (proceza simulado)
1960	JOVIAL	COBOL	ALGOL 60	
	LISP			
1961				APT III (ilmaŝinoj)
				GPSS (simulado)
1962	APL		FORTRAN IV	
	LISP 1.5			
1963			QUICKTRAN	SIMSCRIPT (simulado)
1964		SNOBOL	FORMAC	COGO (konstruado)
			JOSS	
			KLERER-MAY	
1965	PL/1		BASIC	SIMULA (simulado)
1966	APL/360		ALGOL W	CSMP (simulado)
1967			DIAMAG	GRAF (grafika Fortran)
				GSP (grafika konzolo IBM)
1968		SNOBOL 4	ALGOL 68	
			REDUCE	
1969	PASCAL		LEAP	ECAP (elektraj retoj)
				CASSANDRE (elpenso, simulado)
1970				BLISS (uzita por skribo de opso de DEC)
1971			FORDECAL (Formac)	LOGO (instruado)
				PLANNER (robota rego)
			REDUCE 2	
1972	LIS			
1973	APL/SV			LSE (instruado)
				SETL (strukturado de datenoj)
1974	MESA			ALPHARD (opsa skribo)
				CLU (disvolvo)
				TROLL (ekonomikaj modeloj)

jaro	ĝenerala	administra	scienca	aparta
1975	LUCID PROLOG	BAL (baza administro)		
1976	SMALL-TALK LANGAGE C			ATLAS (testo de sistemoj) EUCLID (rego asistata de komputoro : RAK) LASCAR (etendo de Cassandre)
1977	MODULA Z		FORTRAN 77 (V)	
1979	ADA	LCA		ASTEC 3 (elektronikaj maŝaj retoj) CONLAN (priskribo de logikaj sistemoj) LASSO (priskribo de logikaj sistemoj)
1982		DIALOGUE		

Fonto : dictionnaire de l'informatique (Larousse) kompletigita de "Logiciels et Services".

GLOSARO

Adreso : numero kiu identigas memoreron.

Asemblero : programo por traduki programon skribitan en asemblerlingvo en maŝinkodon kaj eble ligi la subrutinojn. La asemblerlingvo, malsame al la kompilerlingvo, dependas de la cela komputoro.

AUO (inkluziva aŭo) : bulea operacio kies rezulto havas la bulean valoron 0 nur se ĉiuj operandoj havas la bulean valoron 0.(vidi KAJO). Aŭ bulea operacio kies rezulto havas la bulean valoron FALSA nur se ĉiuj operandoj havas la bulean valoron FALSA, alikaze la rezulto estas VERA.

Bajto : vico el duformaj elementoj traktata kiel unuo kaj ĝenerale malpli granda ol maŝinvorto.

Bito : unu aŭ la alia el la karaktroj el karaktraro konsistanta el du karaktroj (por ekzemplo: 0 kaj 1).

Bufro : memoro uzata por kompensi aŭ malsamecon de la datenfluo en la diversaj organoj de komputoro aŭ malsamtempecon de okazaĵoj en tiuj organoj, aŭ por enteni nefinajn datenojn.

Bulelogiko, bulea : de la matematikisto Boole. En tiu logiko, la operandoj kaj la rezultoj de ĉiuj operacioj havas nur unu el du eblaj valoroj. (vidi AUO, KAJO).

Buleaĵo : bulea nombro aŭ variablo.

Ĉeno : vico de karaktroj limigita per apartaj limigiloj.

Dateno : reprezento de faktoj, nocioj aŭ instrukcioj laŭ konvencia maniero, taŭga por komuniko, interpreto aŭ trakto far homo aŭ far aŭtomataj iloj.

Debugi : (vidi "erarpelo"). tio estas : serĉi erarojn en programo, trovi ilin kaj modifi la programon por forpeli ilin.

Dekuma, dekera notacio, dekumigo : Dekera notacio uzas dek malsamajn karaktrojn (kutime la dek ciferoj).

Ekzemplo : la karaktra vico 196912312359 povas uziĝi por reprezenti la daton, la horon kaj la minuton de la momento difinita de "unu minuto antaŭ la komenco de la jaro 1970". Tiu ekzemplo uzas dekeran notacion sed ne dekuman notacion, ĉar se mi aldonas 2 al tiu nombro la rezulto estas 197001010001 (unua minuto de la jaro 1970) kaj ne 196912312361 (kiu estas la rezulto en dekuma notacio).

Duuma, duera notacio : vidi difinon de "dekuma ..."

Ekspresio : tio estas frazo kiu indikas komputagojn kaj konsistas el operandoj (t.e. variabloj, konstantoj kaj funkcioj) kaj operatoroj.

Entjero : iu nombro el la senfina vico ....., -3, -2, -1, 0, 1, 2, 3, ...

Erarpelo : laboro post la skribo de programo kaj kompilerado aŭ asemblerado kiu dum la efektivigo de la programo konsistas el serĉo de la eraroj kaj korektado de la programo.

Ico : mallongigo de "Integraj Cirkvitoj" uzata kiel nova radiko. Icujo : skatoleto kun stangetoj aŭ kruroj kiu entenas icon. Icujingo , icuja soklo : materia elemento lutebla al cirkvita tabulo en kiun la teknikisto enŝovas la stangetojn de icujo por konekti ĝin al la tabula cirkvito.

Idli : stato de prilaborilo efektiviganta programon kiam ĝi atendas eksteran okazaĵon por progresi en la efektivigo de la programo.

Implementi : realigi kaj debugi komputoran programon.

Indico : Io (variablo aŭ nombro) kiu indikas rangon de elemento en grupo de elementoj.

Informo (en datena traktado) : signifo kiun homo donas al datenoj, helpe de la konvencioj uzataj por reprezenti ilin.

Informadiko : scienco pri la racia traktado, precipe per aŭtomataj maŝinoj, de la informado rigardata kiel liverilo de la scioj kaj komunikaĵoj en teknika, ekonomika aŭ socia kampo. (PIV).

Inga subrutino : subrutino en kiu ekzistas voko(j) al tiu subrutino aŭ al alia(j) subrutino(j).(vidi "sinvoka subrutino").

Instrukcio : (instrukcikodo, instrukciregistrumo, instrukcimontrilo, instrukciciklo, saltinstrukcio, vokinstrukcio) : en programlingvo, signifoplena esprimaĵo kiu indikas operacion kaj eventuale ties operandojn. Instrukcikodo : kodo uzata por reprezenti instrukcion de instrukciaro. Instrukciregistrumo : registrumo kiu ricevas ĉiujn instrukciojn por ties efektivigo. Instrukcimontrilo : registrumo kiu entenas la adreson de instrukcio.

KAJ : bulea operacio kies rezulto havas la bulean valoron 1 nur se ĉiuj operandoj havas la bulean valoron 1 aŭ kies rezulto havas la bulean valoron VERA nur se ĉiuj operandoj havas la bulean valoron VERA (alikaze la rezulto havas la valoron FALSA).(vidi "AU").

Karaktro : elemento el aro uzata por konsistigi, regi aŭ reprezenti datenojn.

Noto : karaktroj povas esti literoj, ciferoj, interpunktadaj

markoj aŭ aliaj simboloj, ofte reprezentataj per kunmeto de ligitaj aŭ apudaj strekoj aŭ forme de fizikaj statoj en datenujoj.

Kartlegilo : periferia aparato de komputoro kiu ebligas la enigon de datenojn pere de lego de trukartoj.

Kompilero : tiu tradukprogramo por programigolingvo, kiu bazas sur internaciaj interkonsentoj kaj taŭgas por la solvo de apartaj problemoj, ekz. ALGOL por la teknikaj kaj sciencaj, COBOL por la administraj ktp. (PIV). La kompilerlingvo, malsame al la asemblerlingvo estas sendependa de la cela komputoro.

Komputoro : komputilo kapabla efektivigi ampleksan komputadon inkluzivantan multajn aritmetikajn aŭ logikajn operaciojn, sen interveno de homa funkciigisto dum la komputada efektivigo.

Kontekst o : tuto de la cirkonstancoj en kiuj situas fakto kaj kiuj donas al ĝi ĝian valoron kaj ĝian signifon.

Logika operacio : operacio en kiu ĉiu karaktro de la rezulto dependas el maksimume unu karaktro el ĉiuj operandoj. (vidi "AU", "KAJ").

Materialo (de datentraktado) : tuto aŭ fizika elemento en datentraktado, kontraste al asociitaj programoj, proceduroj, reguloj kaj dokumentaro.

Mikroprocesoro : elektronika elemento (ico) kapabla efektivigi instrukciojn.

Mikrokomputoro : komputoro kies komputorada elemento estas mikroprocesoro.

Moduleo, modulea programado : parto de programo partigita en plurajn partojn kies interno estas sendepende studitaj kaj programitaj.

Moduluso : nombro je kiu la reprezento de la kvanto fariĝas nulo. Por ekzemplo ni disponas pri fako en kiu ni povas meti nur unu ciferon; en ĝi estas 9, se mi aldonas 1, la rezulto estas 10, sed la fako kiu povas enteni nur unu ciferon konservas la dekstran, tio estas 0; tiukaze ni diras ke la moduluso estas 10.

Montrilo, stakmontrilo, instrukcimontrilo : registrumo aŭ variablo kiu enhavas adreson de memorero. Stakmontrilo : entenas adreson de memorero el la stako. Instrukcimontrilo : entenas adreson de memorero enhavanta instrukcion.

Okteto, okbito : okbita bajto. "okteto" estas preferebla al okbito (mallongigo de okbitaĵo, vortfaro kiel trimasto). Tiel ni povas formi vorton por ĉiuj bitgrupoj : duteto, triteto, kvarteto, kvinteto, sesteto, septeto, okteto, ..., n-teto.

Operatoro : simbolo de operacio aplikota al operando(j).

Operando : elemento al kiu aplikiĝas operacio.

Opso : neologismo por "OPerating System". Tuto de la programoj por utiligi komputoron kiel eble plej bone.

Programo : laborplano indikanta plenumendajn agojn por ekhavi la solvon de problemo, reprezentata per formo taŭga por efektivigo per komputoro.

Printero : aparato de komputoro kiu ebligas la eligon de presitaj rezultoj kutime sur zigzage faldita papero.

Reelo : nombro kiu povas uziĝi por difini la pozicion de punkto sur akso.

Regcirkvitaro : elektronika elemento en pli ampleksa elemento kiu ebligas la regon de aro da aliaj internaj elementoj.

Registrumo : memoro kun difinita kapacito, por ekzemplo unu bito, unu okteto aŭ unu maŝinvorto kaj kutime uzata por aparta celo. Registrumparo : du kunmetitaj registrumoj uzataj kiel unuo por aparta celo (adreso, granda kvanto ...). Registrumpanelo : en la prilaborilo, grupo de apartaj registrumoj.

Reiniti, initi : tio estas meti en bone konatan staton ĉe la komenco de funkciado aŭ post detekto de eraro. Tiu ago konsistas el seto de variabloj, videbligo de mesaĝoj ...

Reiro : lasta efektivigata instrukcio en subrutino kiu ebligas daŭrigi la efektivigon de programo tuj post la voko de tiu subrutino en tiu programo.

Ringŝovi, ringpaŝigi: fari operacion pri duera kvanto kiu konsistas el meto de la lasta bito en la unuan pozicion. Por ekzemplo 01101 post dekstra ringŝovo fariĝas 10110 aŭ post maldekstra ringŝovo 11010.

Runi : por programo, tio estas esti en la stato, kiam la prilaborilo efektivigas ĝiajn instrukciojn.

Seti : akirigi al io apartan staton aŭ valoron. Por ekzemplo, por initi oni ofte setas la variablojn je nulo en programo.

Sinvoka subrutino : subrutino en kiu ekzistas voko(j) al tiu subrutino (rekte sinvoka subrutino) aŭ al alia(j) subrutino(j) en kiu(j) ekzistas voko(j) al tiu unua subrutino (nerekte sinvoka subrutino).

Skalaro : nombro kiu difiniĝas per sia mezuro, ĝi povas esti entjero, reelo aŭ buleaĵo.

Softvaro : tuto de la programoj, proceduroj, reguloj kaj asociigita dokumentaro rilata al la funkciado de datentraktada sistemo.

Stako : memoro kies datenoj estas traktataj tiel ke la unua havebla estas la laste enmemorigita ankoraŭ en la memoro. Tio funkcias kiel LENUELa vico (Lasta EN, Unua EL).

Stori, enmemorigi : enigi datenon en memoron.

Subrutino : programo aŭ parto de programo kiu povas havi ĝeneralan aŭ oftan uzon.

BIBLIOGRAFIO

(1) : The programming language PASCAL. N. WIRTH. Acta Informatica 1, 35-63. (1971).

(2) : An axiomatic definition of the programming language PASCAL. C.A.R. HOARE and N. WIRTH. Acta Informatica 2, 335-355. (1973).

(3) : PASCAL, user manual and report. K. JENSEN and N. WIRTH. Springer-Verlag ed. (1975).

(4) : Le langage de programmation PASCAL. Spécifications et mise en oeuvre. 2e édition. AFNOR-SOL. (1982).

(5) : Introduction au PASCAL. Pierre LE BEUX. SYBEX. (1980).

(6) : MOTOROLA PASCAL language manual. (1979).

Tabelo de la enhavo

De la sama aŭtoro:

Artikoloj :

- Teledateniko kaj novaj produktoj. Interligilo Nro 66 (2). Marto-Aprilo 1979. p8-11.
- Intel 8080 - Priskribo kaj programado. Scienca Revuo. Vol 30. Nro 1 (133): Teĥnika kajero. 1979. p7-38.
- Konvertilo de framnivela proceduro kun paknivelo konforma al la Rekomendo X25 de CCITT. ISU de UEA-kongreso en Lucerno Svislando. 1979. 15p.
- Elektronika jarlibro. Monato 4/80. p30.
- La submaraj kabloj. Interligilo Nro 80 (4) Julio-Aŭgusto 1981, la parto. p8-10. Nro 78 (2) Marto-Aprilo 1981, 2a parto. p10-11.
- La plurcela ligilnivela proceduro PLP. Unua Simpozio pri Komputiko. Rennes Francujo. Julio 1981. p194-203.
- PACKSATNET: la indonezia pakkomuta datentransiga reto. INTERKOMPUTO 82. Budapest Hungario. decembro 1982.

Kajeroj aŭ libroj:

- La datentransiga proceduro X25 kun ĝenerala enkonduko pri proceduroj. Verkinto. SUK KJ04. 1981. 133p.
- Unua simpozio pri komputiko. Rennes Francujo. Julio 1981. Redaktoro. 19 kontribuaĵoj. 269p.
- La mikroprocesoro INTEL 8080. Tradukinto. 1982. 77p.
- La programlingvo PASCAL. Verkinto. 1982. 72p.